NASA GSFC Advanced Architectures and Automation (Code 588)

SPLAT SOFTWARE DESIGN DOCUMENT

July 12, 2002

Software Design Document for SPLAT

Prepared by:

Jeffrey M. Robinson

Aquilent, Inc.

Document Summary

Document Title	Software Design Document for SPLAT
Author	Jeffrey Robinson
Status	Draft

Document Change History Log

Date of Change	Version	Summary of Change
June 12, 2002	1.0	Initial Release
June 17, 2002	1.1	Made MMS Interface TBD
June 18, 2002	1.2	Modifications per internal review

Approvals

Title	Name	Signature	Date

Table of Contents

List of Figures

1 Introduction

The purpose of this document is to describe the software design for the SSR Playback Automation Tool (SPLAT) formerly Goal Oriented Commanding (GOC) being developed at the NASA Goddard Space Flight Center Advanced Automation and Architectures Branch (Code 588). The document describes SPLAT and will be used to drive the design and implementation of the system.

2 References

The following references were used in preparation of this document:

1. SPLAT System Requirements Specification version 1.2 April 18, 2002.
2. SPLAT Software Architecture Document version 1.3 June 17, 2002.

3 Architectural Representation

This document presents the system architecture as a series of views: A Logical View and a Process View. These views are presented as Together Models that use the Unified Modeling Language (UML).

The Logical View of the architecture describes the most important classes. Class diagrams are included to illustrate the relationships between architecturally significant classes and other elements within the system

The Process View describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. It also describes the allocation of objects and classes to tasks.

4 Architectural Goals & Constraints

This section enumerates key requirements and system constraints that have a significant bearing on the architecture. They are:

1. The architecture should support retrieving report data either from the MMS system automatically or via manual placement of input files in a common directory on the operator's local machine (PC).
2. The primary user interfaces for playback scheduling, etc. must be able to run on the on a user's Java enabled PC.
3. All usability, reliability, performance and loading requirements as stipulated in the SPLAT System Requirements Document [1], must be taken into consideration as the architecture is being developed.

5 Logical View

In the Logical View the collaborations detailed in the Software Architecture Document's Use Case View [2] are combined into class diagrams that depict the significant architectural elements. These are then organized into packages and service layers to create an Architectural Overview. The results of an analysis of concurrency and inter-process communication requirements are presented in the Process View.

5.1 Packages and Subsystems

The SPLAT system is organized into three layers. The User Services Layer provides user interfaces for workflow management and interaction with schedules. The Scheduling Services Layer encapsulates the processing required to ingest and translate MMS scheduling reports, and create SSR buffer playback schedules, and the Data Management Layer. The packaging of classes within these layers is described in the following sections.

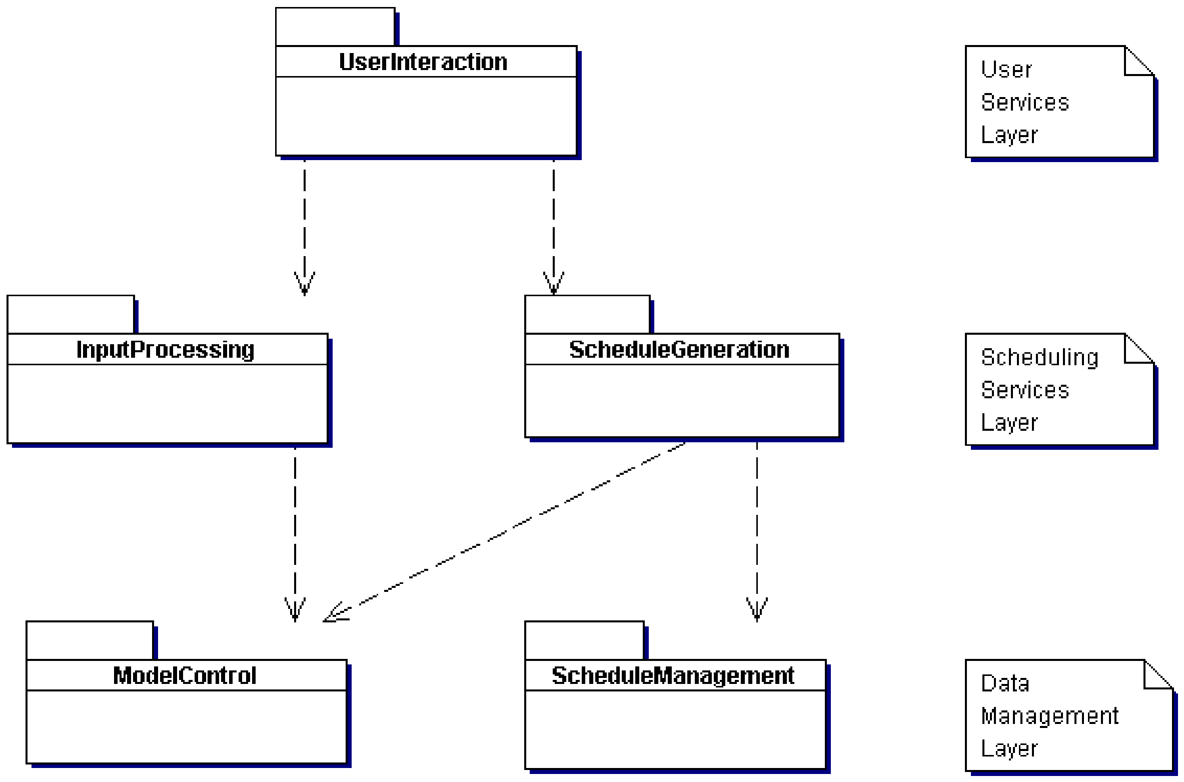


Figure 5-1: Packages and Layering for SPLAT

5.1.1 User Interaction

The User Interaction package contains the GUI components and supporting classes that allow the user to enter special event scheduling windows, select print and display filters, view the playback scheduler, and control schedule generation. These include:

?? ASTER Rates UI
?? Main Window
?? Print Schedule UI
?? Save Schedule UI
?? Parameter UI
?? Filter UI
?? Dump Windows UI
?? Sync Point Parameters UI
?? Scheduling Options UI
?? Input Reports UI

5.1.2 Input Processing

The Input Processing package contains the classes and interfaces required to ingest and manage the input report files from the MMS (TBD) or a local directory, extract the contacts and MODIS/MISR mode change events from them. It includes the:

?? Report Manager
?? Event Contact Manager
?? Report List
?? Buffer States
?? MMS Interface (TBD)
?? Playback Windows

5.1.3 Schedule Generation

The Schedule Generation package contains the classes that control and execute creation of SSR buffer playback schedules. The included classes are:

?? Schedule Manager
?? Dump Windows
?? Sync Point Parameters

5.1.4 Model Control

The Model Control package contains classes that provide information describing modeling parameters and modes of operation for the Terra instruments as well as scheduling options affecting operation of the tool. These include:

?? Scheduling Options
?? Modeling Parameters

5.1.5 Schedule Management

The Schedule Management package contains classes that provide control over schedule storage and display. These include:

?? Event Filters
?? Print Display Filters
?? Playback Schedule

5.2 Design Model

This section describes in greater detail the classes identified as architecturally significant in the Software Architecture Document [2]. The classes are documented through a series of class diagrams describing the classes themselves, and a series of sequence diagrams depicting interactions between the architecturally significant classes and other elements in the system such as User Interfaces.

5.2.1 Class Diagrams

This section contains class diagrams for the architecturally significant classes of the SPLAT system and descriptions of the major methods for each class. Note that not all methods are mentioned here. For the sake of brevity, descriptions for get and set methods are omitted.

Note that persistent data storage for the SPLAT system is being provided by the Java Application Shell (JAS). Class diagrams are not provided for JAS features.

5.2.1.1 Buffer States

The Buffer States class contains information for a single buffer state entry. This information represents a Buffer State entry extracted from the SSR Buffer States Report. Each Buffer State entry contains the current usages and durations for the SSR's ASTER, MISR and MODIS buffers. The three entries together make up the contents a Buffer State entry and represent the fullness of each of the SSR buffers at a given time. The Buffer States are used during scheduling to determine the Synchronization Point.

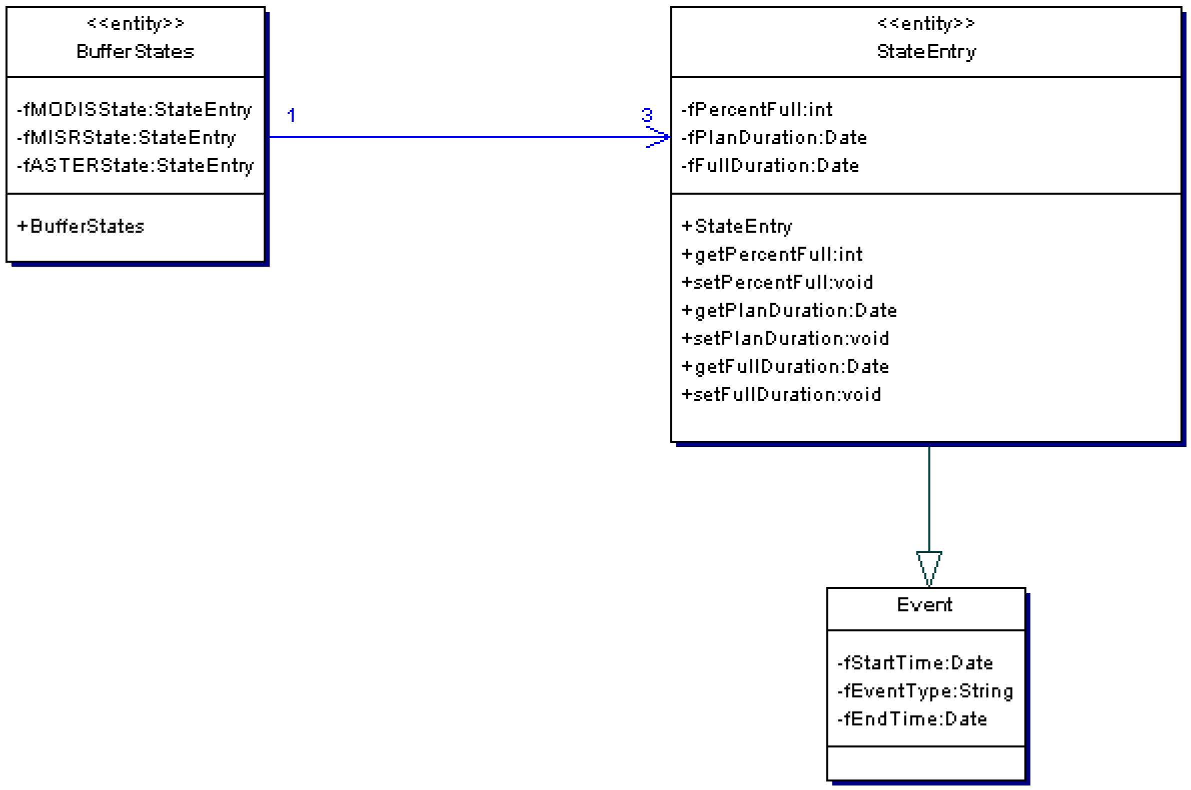


Figure 5-2: BufferStates Class Diagram

5.2.1.1.1 Major Methods

None

5.2.1.2 Contact Window

The Contact Window class contains information corresponding to an individual contact period. Contact Periods are areas of opportunity for dumping the contents of the SSR Buffers. For the SPLAT system, Contact Window entries represent K and S band contacts extracted from the TDRS Contact Report and X band contacts extracted from the Ground Network (GN) Report.

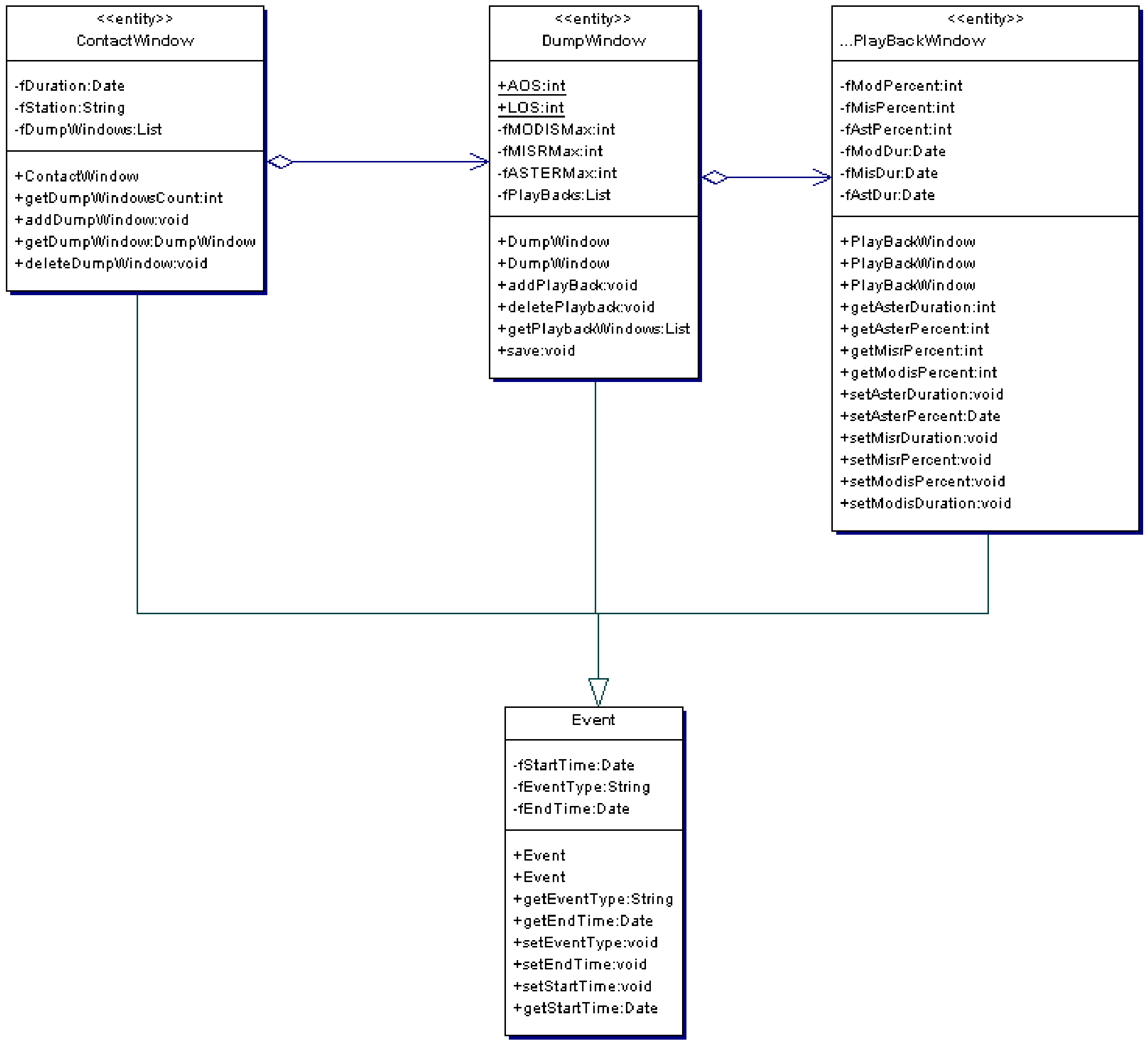


Figure 5-3: ContactWindow class diagram

5.2.1.2.1 Major Methods

?? ***public int getDumpWindowCount()*** – returns the number of dump windows associated with the contact period.

?? ***public int addDumpWindow(Dumpwindow dump)*** – adds the specified dump window to the contact period.

?? ***public int deleteDumpWindow(DumpWindow dump)*** – removes the specified Dump Window from the contact period.

5.2.1.3 Dump Windows

The Dump Window class contains the information necessary to describe a single dump window. Dump windows represent areas of opportunity within a contact period for playback of the SSR buffers. Each ContactWindow contains at most two dump windows per contact. Each dump window can have an unlimited number of playback windows. Note that the details of the playback window class are not described in this diagram, nor is the relationship with the ContactWindow class. The intent here was to represent the relationship between a Dump Window and it’s associated Playback Windows. For more information regarding the Contact Window class refer to Section 5.2.1.2. For more information on the Playback Window class pleaser refer to Section 5.2.1.8.

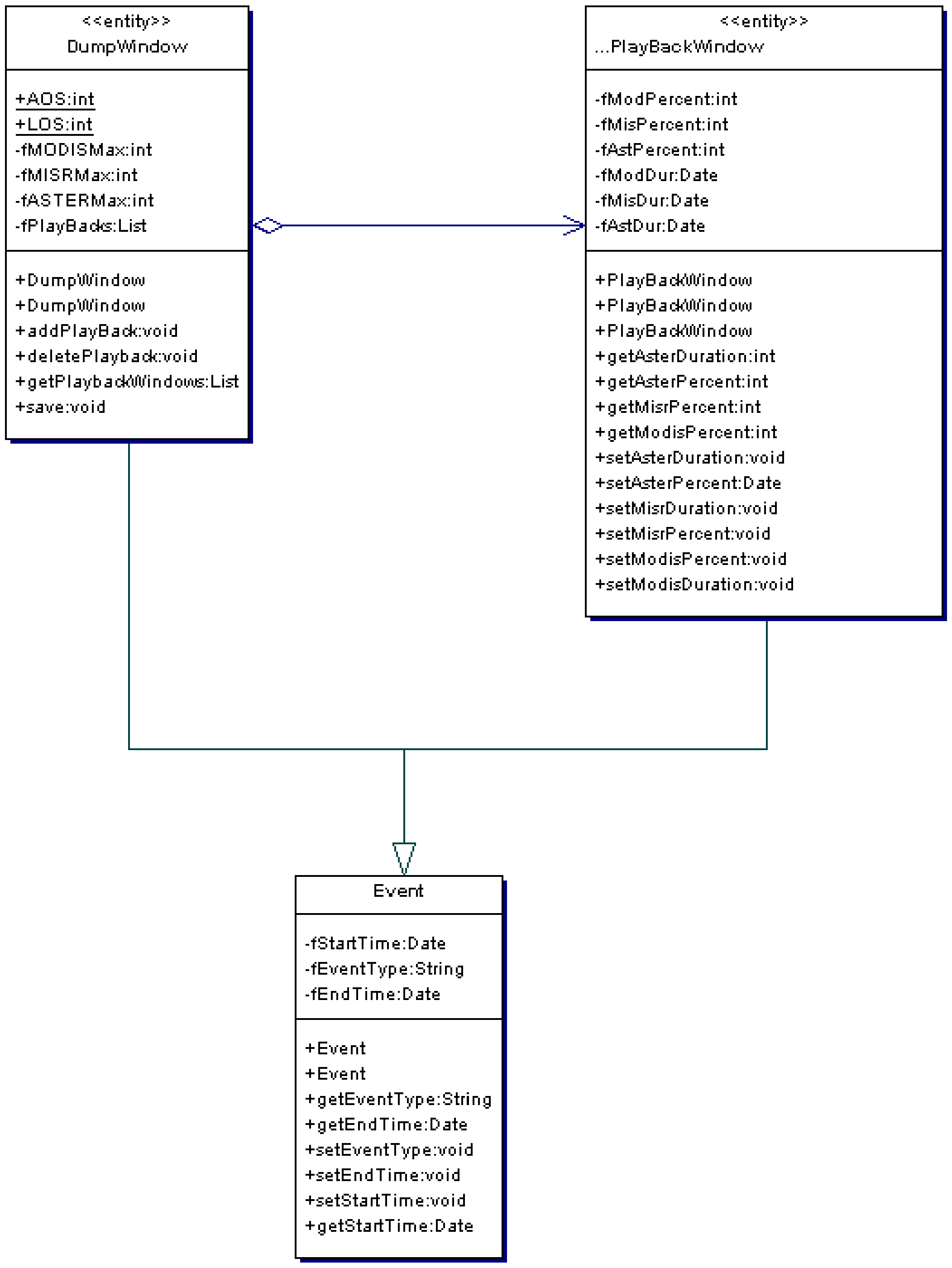


Figure 5-4: DumpWindow class diagram

5.2.1.3.1 Major Methods

?? ***public int addPlayback(PlaybackWindow playback)*** – adds the specified playback window to the list of playbacks for this DumpWindow.

?? ***public int deletePlayback(PlaybackWindow playback)*** – deletes the specified playback window from the current dump window.

?? ***public List getPlaybackWindows()*** – returns a List of playback windows for the dump window.

5.2.1.4 Event Contact Manager

The EventContactManager is a data repository whose main responsibility is maintaining a list of extracted events containing all contact periods, MODIS and MISR mode change events, ASTER modes, and buffer states extracted from the TDRS Contact report, GN Report, SSR Buffer states report, and ATC Load Report. The ReportManager control process initiates, monitors, and controls report parsing and data extraction. The ReportMonitor initiates file parsing (data extraction) when the SSR Scheduler selects the “create contact list” option from the MainWindow. For each available input report, the ReportManager process sends a notification to the EventContactManager identifying the file to be parsed. The EventContactManager then parses the input file extracting the contact periods, mode change events or buffer states entering them into local contact period, mode change, or buffer states lists. Once parsing is complete, the EventContactManager returns the parsing status of the input file to the ReportManager along with the extracted contact periods and dump windows.

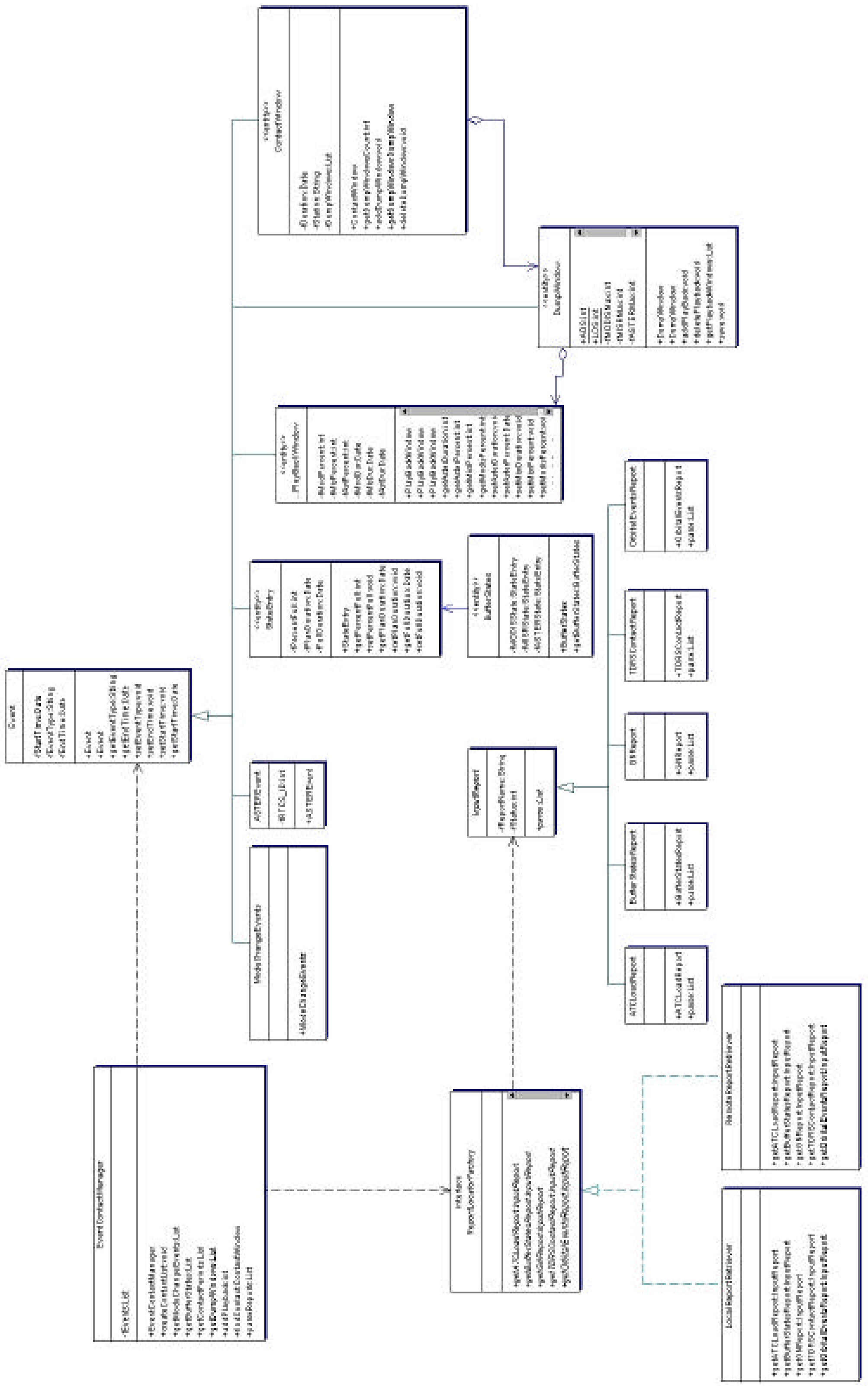


Figure 5-5: EventContactManager class diagram

5.2.1.4.1 Major Methods

EventContactManager

?? ***public int createContactList()*** – creates the list of contact periods. This method actually controls all report parsing.

?? ***public List getBufferStates()*** – retrieves a list of buffer state events extracted from the SSR Buffer States Report.

?? ***public List getModeChangeEvents()*** – retrieves a list of MISR and/or MODIS mode change events.

?? ***public List getContactPeriods()*** – retrieves a list of ContactWindows extracted from the TDRS Contact Report and/or GN Report.

?? ***public List getDumpWindows(po***:ContactWindow) – retrieves a list of DumpWindows for the specified ContactWindow.

?? ***Public ContactWindow findContact***(***Date startTime, String eventType)*** – locates the ContactWindow of the corresponding type and with the specified start time.

?? ***public void addPlayback(ContactWindow contact, DumpWindow dump)*** – adds a new PlaybackWindow to the specified Contact and Dump Window.

?? ***public List parseReports()*** – coordinates report parsing and event/contact extraction for each of the input reports.

LocalReportRetriever and RemoteReportRetriever

?? ***public InputReport getATCLoadReport()*** – retrieves the ATC Load Report for processing. The ATC Load Report contains ASTER imaging events.

?? ***public InputReport getBufferStates()*** – retrieves the SSR Buffer States Report for processing. The SSR Buffer States Report contains buffer full percentages for each of the SSR buffers at contact points.

?? ***public InputReport getOrbitalEventsReports()*** – retrieves the Orbital Events Report for processing. The Orbital Events Report contains the Nadir Term day/night crossing events.

?? ***public InputReport getGNReport()*** – retrieves the Ground Network Report file for processing. The GN Report contains X-band (ground) contacts.

?? ***Public InputReport getTDRSContactReport()*** – retrieves the TDRS Contact Report for processing. The TDRS Contact Report contains K-band and S-band contact periods.

ATCLoadReport

?? ***public List parse()*** – parse routine specific to the ATC Load Report. This method will parse (extract) the ASTER Imaging Events from the ATC Load Report.

BufferStatesReport

?? ***public List parse()*** – parse routine specific to the SSR Buffer States Report. This method will extract the SSR Buffer State entries from the Buffer States Report.

GN Report

?? ***public List parse()*** – parse routine specific to the Ground Network Report. This method will extract X-band (ground) contact entries from the GN Report.

Orbital Events Report

?? ***public List parse()*** – parse routine specific to the AM1 Orbital Events Report. This method parses the MODIS and MISR day/night events from the report.

TDRS Contact Report

?? ***public List parse()*** – parse routine specific to the TDRS Contact Report. This method parses the TDRS contact report, extracting K and S band contact periods.

5.2.1.5 MMS Report Retriever

The MMS Report Retriever class represents the SPLAT interface to the Mission Management Software (MMS) system. This class is TBD and this section is simply a placeholder for future development.. Directly interfacing to the MMS System has been deferred until FY03.

5.2.1.5.1 Major Methods

N/A

5.2.1.6 Modeling Parameters

The Modeling parameters are composed of those values used during schedule determination that affect schedule generation. These values include imaging rates for the individual instruments (ASTER, MISR, MODIS), playback rates for TDRS (K-band) and Ground (X-band) contacts, ASTER imaging modes, buffer capacities and the alike. These values are persistent entities in the SPLAT Tool and will remain from SPLAT session to session. The Modeling parameters are being handled by the Java Application Shell (JAS) constructs and thus are not modeled with a class diagram. For more information regarding the modeling parameters, please refer to the Section 6.6.2.

5.2.1.7 Playback Schedule

The PlaybackSchedule class represents the completed SSR playback schedule. It contains all contacts, dump windows, playback windows, buffer states, mode changes, and comments generated and/or added to a schedule. Note that the values stored in this class represent the full unfiltered schedule. Hardcopy schedules and the data displayed on the MainWindow timeline are based on the data in this class.

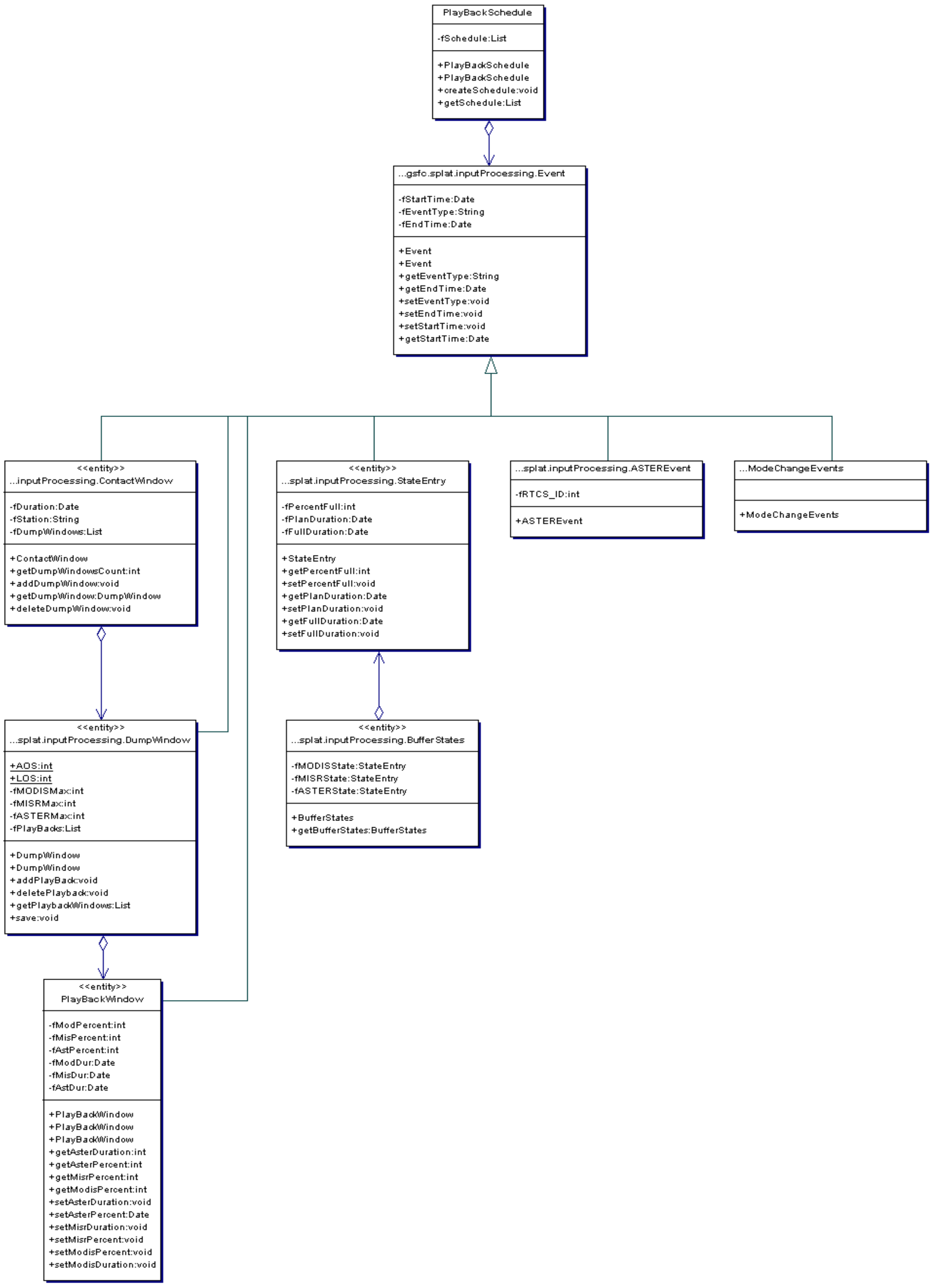


Figure 5-6: PlaybackSchedule class diagram

5.2.1.7.1 Major Methods

?? ***public int createSchedule()*** – creates the playback schedule for display and hardcopy.

?? ***public List getSchedule()*** – retrieves the playback schedule.

5.2.1.8 Playback Windows

The PlaybackWindow class contains all data necessary to represent a single playback window for the SSR buffers. Playback windows are created during schedule generation by the ScheduleManager control class. The placement and settings of an individual playback window are determined based on the dump windows locations, mode changes, and buffer usage.

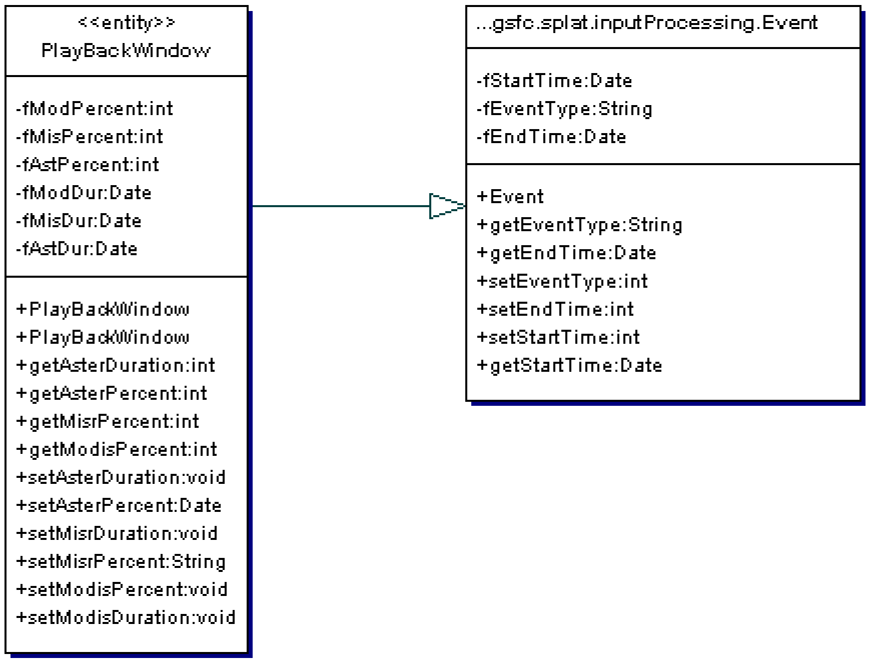


Figure 5-7: PlaybackWindow class diagram

5.2.1.8.1 Major Methods

None

5.2.1.9 Print Display Filters

The Print and Display Filters classes contain information regarding the printable and/or visible fields and event types. Through these classes, the SPLAT user can customize the display of data on the timeline and in hardcopy output. Note that the Print Filters and Display Filters classes are separate instantiations of the Filter class presented here. The Filter class contains entries for both event filters and field filters.

Note that the list of all possible filterable fields in SPLAT is static and will not change. However, the list of filterable event types is dynamic and will change from SPLAT session to session depending on which reports are used for schedule generation.

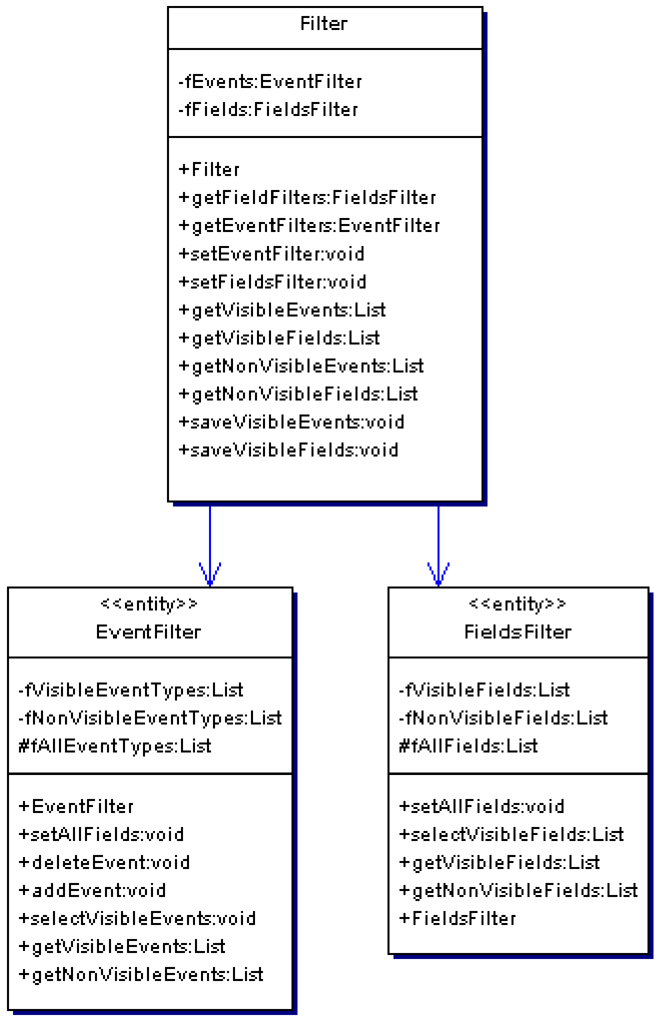


Figure 5-8: Filter class diagram

5.2.1.9.1 Major Methods

EventFilter

?? ***public int deleteEvent(EventType event)*** – removes the specified event from the list of available events to filter.

?? ***public void addEvent(EventType event)*** – adds the specified event to the list of filterable event types.

?? ***public void selectVisibleEvents()*** – updates the visible events list to reflect the user selected visible event types.

?? ***public List getVisibleEvents()*** – retrieves the list of visible event types.

FieldsFilter

?? ***public void selectVisibleFields()*** – updates the visible field list to reflect the user selected visible fields.

?? ***public List getVisibleFields()*** – retrieves the list of visible fields.

5.2.1.10 Report List

The ReportList class maintains the names and locations of the input reports used during schedule generation. A separate String is maintained for each of the possible input reports. Each String contains both the report name and location (directory path).

Note that schedule generation requires a minimum of three reports (TDRS Contact Report, Orbital Events Report, and SSR Buffer States) and a maximum of five reports (TDRS Contact Report, Orbital Events Report, SSR Buffer States Report, GN Report, ATC Load Report) depending on the options selected.

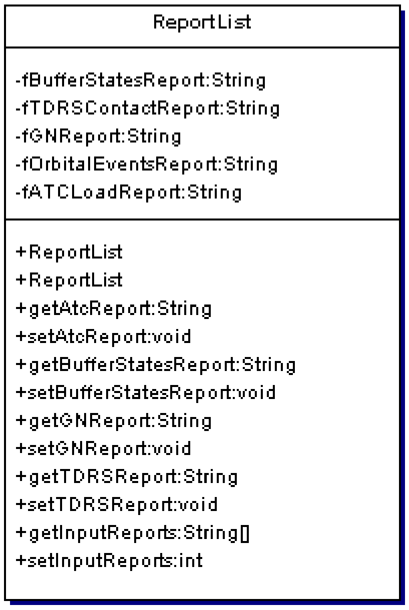


Figure 5-9: ReportList class diagram

5.2.1.10.1 Major Methods

?? ***public String[] getInputReports()*** – retrieves an Array of Strings containing the names and locations of the necessary input reports.

?? ***Public int setInputReports(String[] reports)*** – sets the input report names and locations to the values in the specified array of Strings.

5.2.1.11 Report Manager

The ReportManager control class is responsible for controlling all aspects of report ingestion and parsing. This includes parsing the input reports, setting the filterable events types, determining the initial dump windows, locating the synchronization point candidates and selecting an initial sync point.

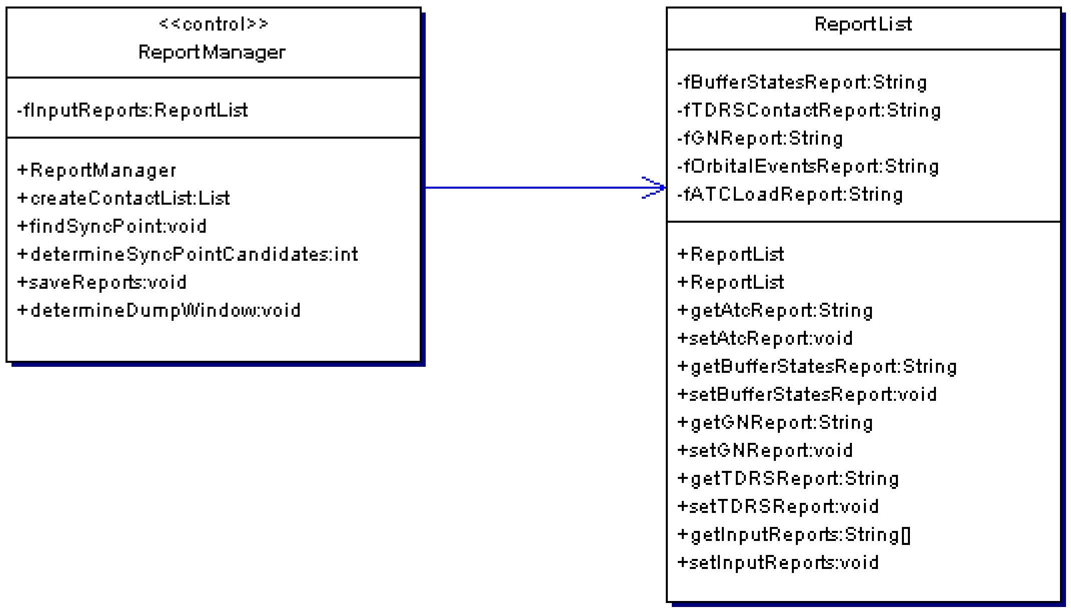


Figure 5-10: ReportManager class diagram

5.2.1.11.1 Major Methods

?? ***public List createContactList()*** – coordinates and manages report parsing and construction of the event and contact list. The list of contacts, etc is returned after execution completes.

?? ***public void findSyncPoint()*** – coordinates and controls selection of the synchronization point.

?? ***public int determineSyncPointCandidates()*** – coordinates and controls the operation of determining the sync point candidates from the buffer states entries.

?? ***public int saveReports()*** – saves the names and locations of the reports retrieved manually from MMS in the ReportList.

?? ***public void determineDumpWindows()*** – coordinates and controls the automated Dump Window determination process.

5.2.1.12 Schedule Manager

The ScheduleManager control class is responsible for controlling and coordinating all operations that affect the Playback Schedule. The Schedule Manager provides support for generating the playback schedule, filtering the schedule for printing, printing the schedule, and saving the schedule to a file.

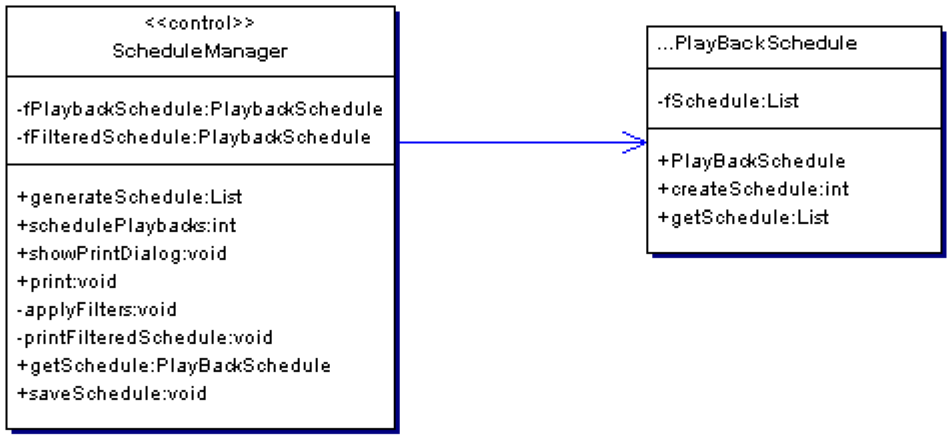


Figure 5-11: ScheduleManager class diagram

5.2.1.12.1 Major Methods

?? ***public List generateSchedule()*** – creates the playback schedule for display and/or printing.

?? ***public int schedulePlaybacks()*** – creates playback windows for each of the dump windows in the scheduling horizon.

?? ***public void showPrintDialog()*** – raises the print dialog window in response to the user selection of the print schedule option.

?? ***public void print()*** – “prints” the playback schedule to either a local file or to a printer.

?? ***public void applyFilters()*** – filters the schedule data based on the Event and Field print filter settings.

?? ***public PlaybackSchedule getSchedule()*** – retrieves the full unfiltered playback schedule.

?? ***public void saveSchedule()*** – saves the full unfiltered playbacks schedule to a file.

5.2.1.13 Scheduling Options

The SchedulingOptions class contains user provided information regarding the scheduling window, report locations, etc. The information contained in this class includes: the scheduling window start and stop times; report locations, the ASTER modeling mode, etc. These options are used during report parsing and schedule generation.

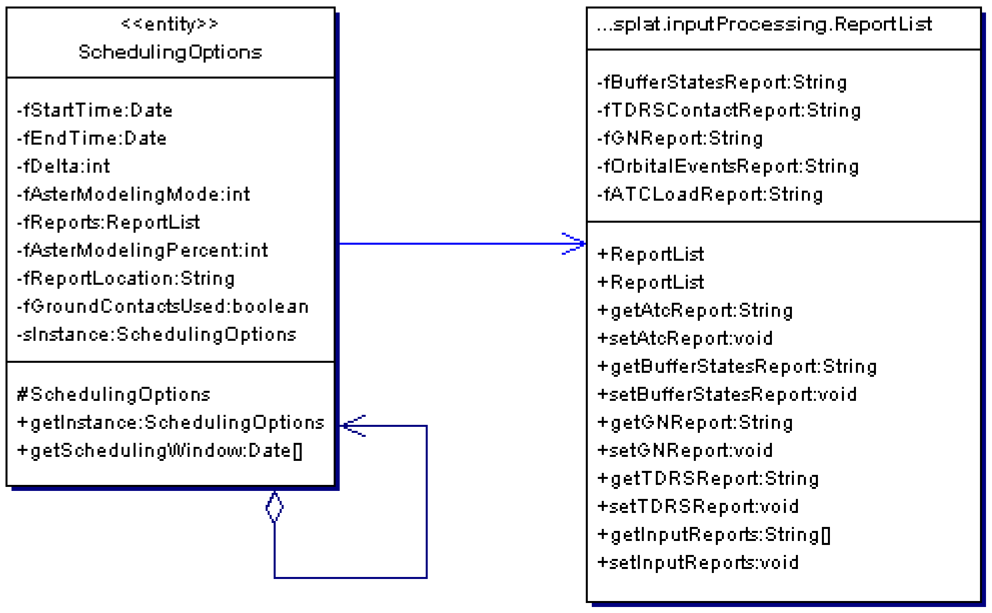


Figure 5-12: SchedulingOptions class diagram

5.2.1.13.1 Major Methods

?? ***public SchedulingOptions getInstance() –***retrieves the single instance of the schedule options class.

?? ***public Date[] getSchedulingWindow()*** – returns the scheduling window start time, end time, and delta in a Date array.

5.2.1.14 Sync Point Parameters

The SyncPointParameters class is responsible for maintaining the list of synchronization points candidates for the delta window prior to the start of the scheduling period. Each Sync Point entry contains the information for a candidate synchronization point and includes a time stamp, the buffer capacities, and the buffer durations for the ASTER, MISR, and MODIS buffers. From these candidate entries the SPLAT tool selects a single Sync Point Parameter entry as the synchronization point. The synchronization point represents the latest contact before the start of the scheduling window in which the SSR buffers can be completely emptied.

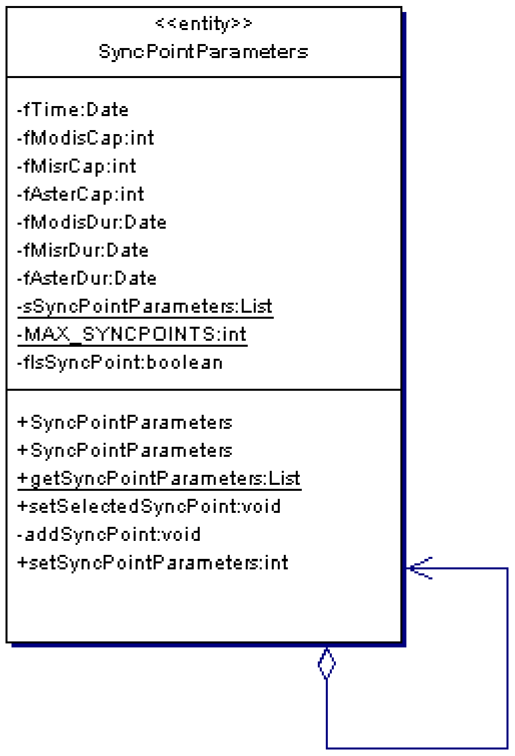


Figure 5-13: SyncPointParameters class diagram

5.2.1.14.1 Major Methods

?? ***public List getSyncPointParameters() –*** retrieves a list of all candidate synchronization point entries.

?? ***public void setSyncPointParameters(List parms) –*** sets the values of an individual sync point entry to those specified by the parms argument.

?? ***public int selectSyncPoint(SyncPointParamaters point) –*** selects the specified sync point candidate as the synchronization point.

?? ***public void addSyncPoint(SyncPointParameters point) –*** adds the specified sync point entry to the list of candidate Synchronization point entries.

5.2.2 Sequence Diagrams

This section contains sequence diagrams that describe the interactions between classes and components within the SPLAT Tool. The sequence diagrams are separated by system function. Sequence diagrams are provided for report processing, schedule generation, automated dump window determination, and automated sync point determination.

Note that persistent data storage for the SPLAT system is being provided by the Java Application Shell (JAS). Sequence diagrams are not provided for JAS features.

5.2.2.1 Determine Dump Windows Sequence Diagram

Figure 5-14 describes the interactions between classes during dump window determination. The automated dump window determination process involves the SPLAT tool automatically selecting the initial set of dump windows based on the contact periods extracted from the Input Reports.

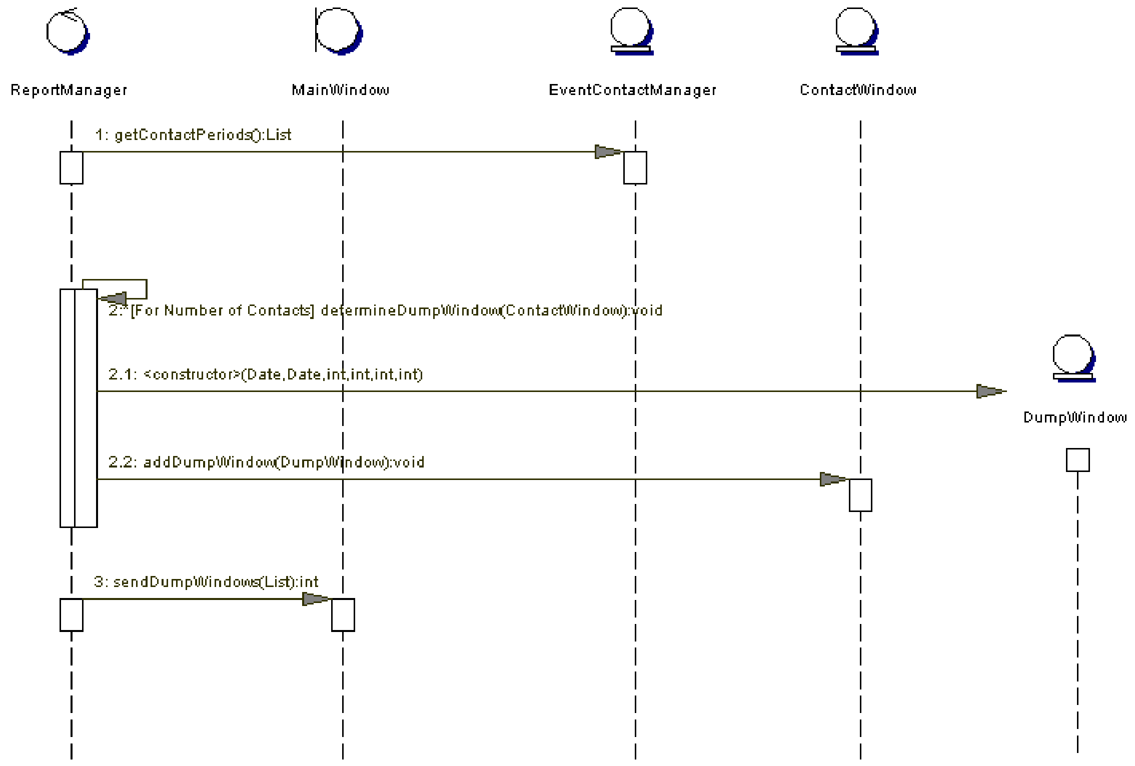

ReportManager
MainWindow
EventContactManager
ContactWindow
DumpWindow
1: getContactPeriods():List
2:*[For Number of Contacts] determineDumpWindow(ContactWindow):void
2.1: <constructor>(Date,Date,int,int,int,int)
2.2: addDumpWindow(DumpWindow):void
3: sendDumpWindows(List):int

Figure 5-14: Find Dump Windows

5.2.2.2 Determine Sync Point Sequence Diagram

Figure 5-15 describes the interactions between classes during synchronization point determination. Sync point determination involves the SPLAT tool automatically selecting the synchronization point candidates from the delta window and an initial synchronization point.

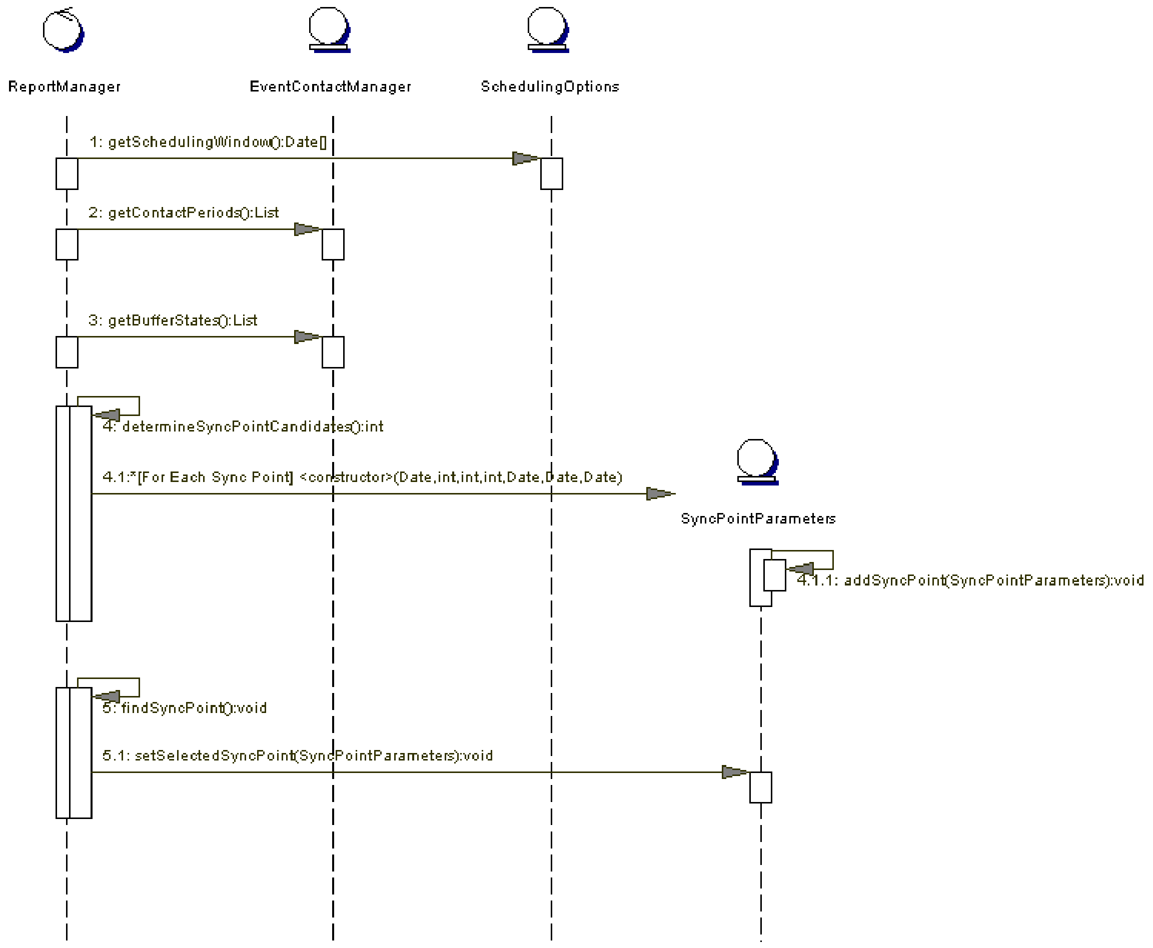

ReportManager
EventContactManager
SchedulingOptions
1: getSchedulingWindow():Date[]
2: getContactPeriods():List
3: getBufferStates():List
4: determineSyncPointCandidates():int
4.1:*[For Each Sync Point] <constructor>(Date,int,int,int,Date,Date,Date)
SyncPointParameters
4.1.1: addSyncPoint(SyncPointParameters):void
5: findSyncPoint():void
5.1: setSelectedSyncPoint(SyncPointParameters):void

Figure 5-15: Determine Sync Point

5.2.2.3 Generate SSR Buffer Playback Schedule

Figure 5-16 describes the interactions between classes during playback schedule generation. Playback schedule generation involves determining the playback window locations and durations for each of the dump windows based on buffer usage and mode changes.

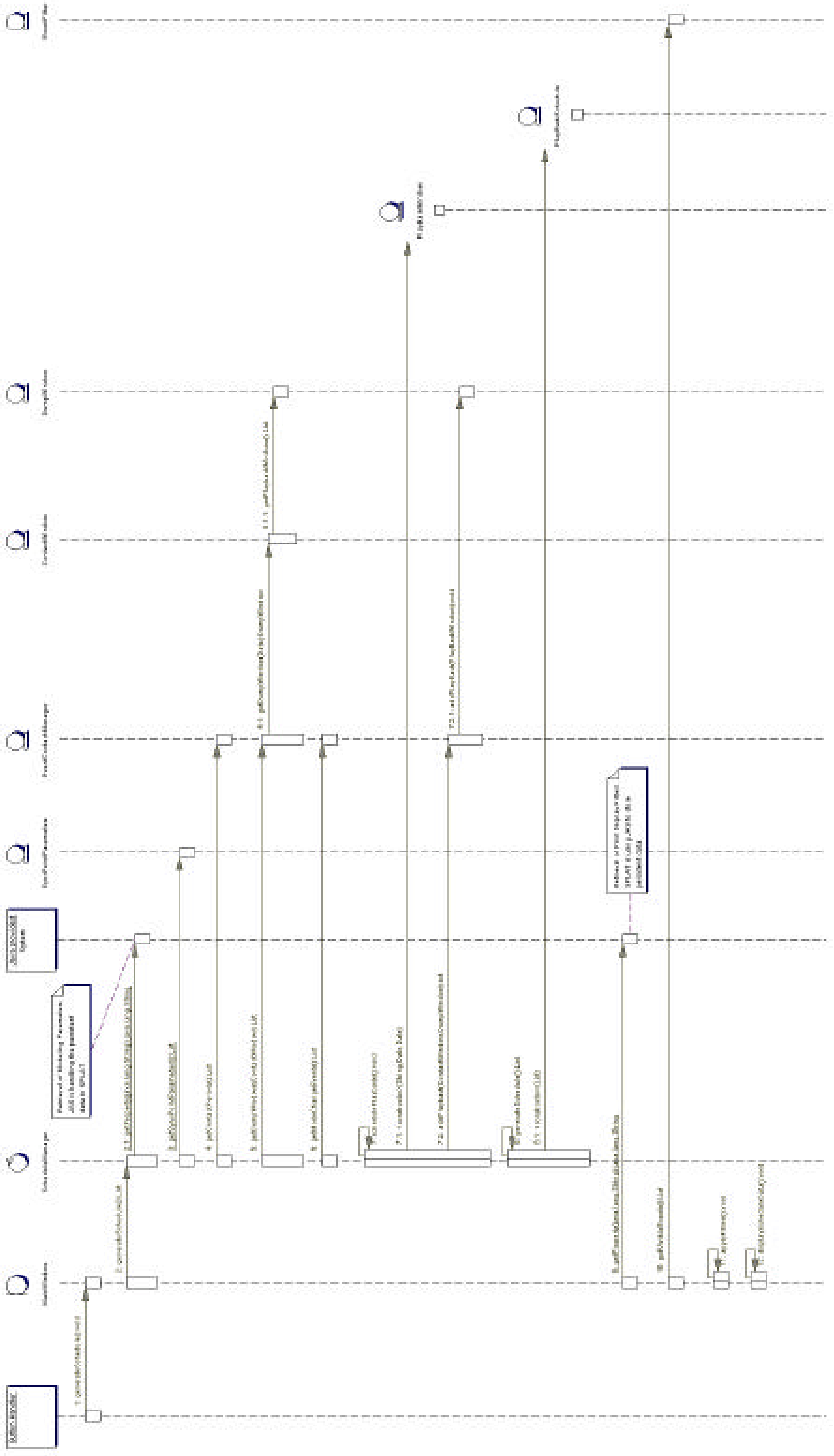


Figure 5-16: Generate Playback Schedule

5.2.2.4 Process Input Reports Sequence Diagram

Figure 5-17 describes the interactions between classes during report processing. Report processing involves retrieving and parse the individual input reports, extracting mode changes, contact periods, orbital events, and ASTER imaging events.

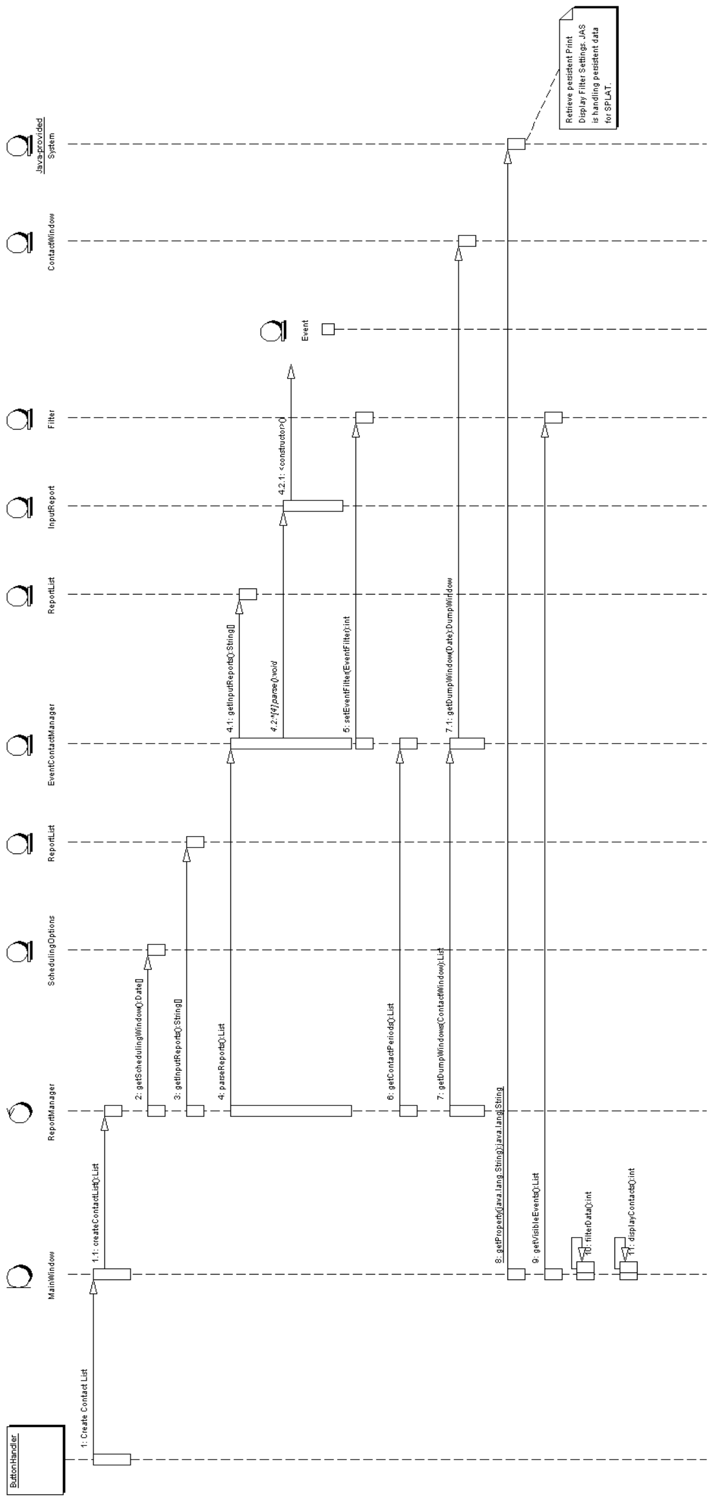

ButtonHandler
MainWindow
ReportManager
SchedulingOptions
ReportList
EventContactManager
ReportList
InputReport
Filter
Event
ContactWindow
Java-provided
System
1: Create Contact List
1.1: createContactList():List
2: getSchedulingWindow():Date[]
3: getInputReports():String[]
4: parseReports():List
4.1: getInputReports():String[]
4.2: *[4] parse():void
4.2.1: <constructor>()
5: setEventFilter(EventFilter):int
6: getContactPeriods():List
7: getDumpWindows(ContactWindow):List
7.1: getDumpWindow(Date):DumpWindow
8: getProperty(java.lang.String):java.lang.String
9: getVisibleEvents():List
10: filterData():int
11: displayContacts():int
Retrieve persistent Print Display Filter Settings. JAS is handling persistent data for SPLAT.

Figure 5-17: Process Input Reports

6 User Interface Design

This section includes descriptions and class diagrams for each of the GUI components in the SPLAT system. For each GUI, a class diagram depicting the GUI components and one or more sequence diagrams depicting interaction with other classes in the system is provided.

6.1 ASTER Rates UI

The ASTERRatesUI is a dialog through which the user can modify existing ASTER imaging rates or add new imaging modes. Through this interface the user can select the name of the modeling mode, the Real Time Command Sequence ID number (RTCS_ID) for the mode, which instruments are active, and when the instruments turn on or off.

6.1.1 ASTER Rates UI Class Diagram

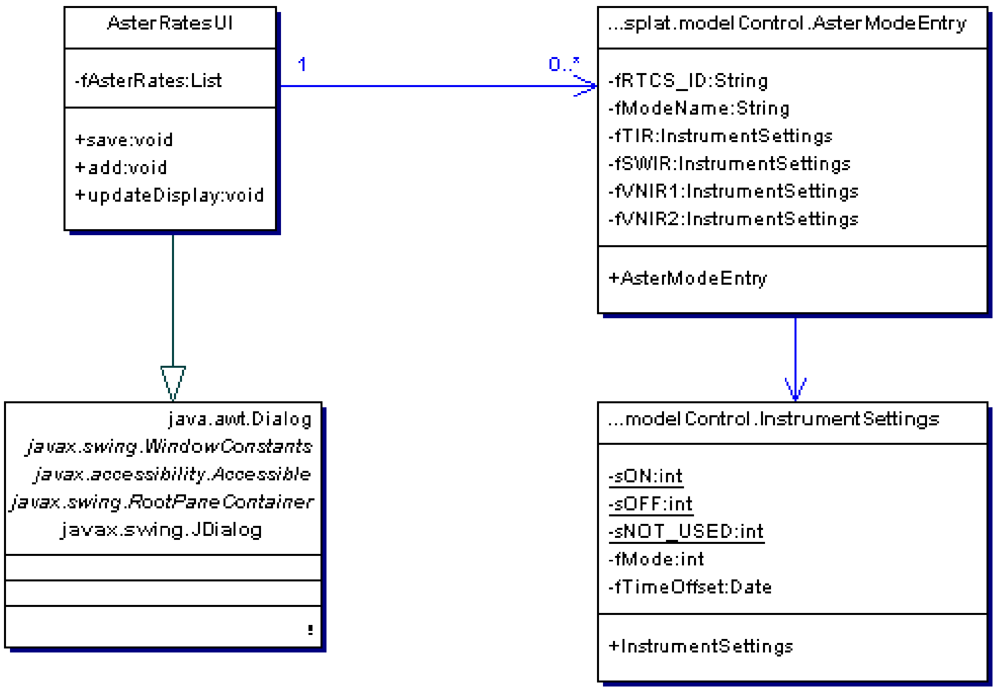


Figure 6-1: ASTERRatesUI class diagram

6.1.1.1 Major Methods

- ***public void save()*** – saves the modified or new ASTER modes of operations.
- ***public void add()*** – creates a new ASTER Mode and adds it to the list of modes.
- ***public void updateDisplay()*** – updates the ASTER Rates UI display to reflect added or modified ASTER mode entries.

6.1.2 Add an ASTER Mode Sequence Diagram

Figure 6-2 describes the interactions between classes when creating a new ASTER imaging mode.

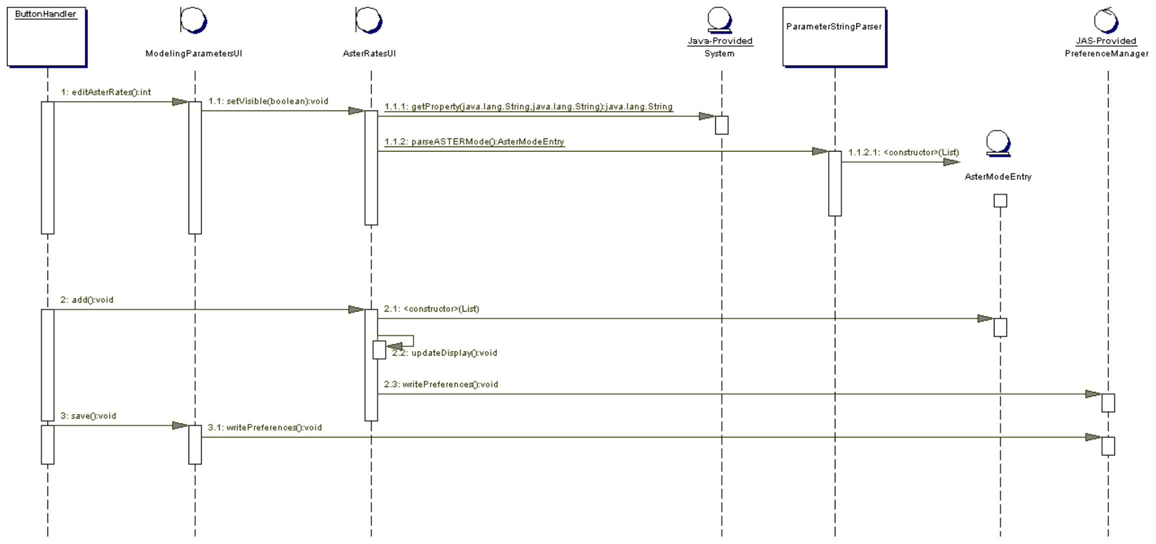

ButtonHandler
ModelingParametersUI
AsterRatesUI
Java-Provided System
ParameterStringParser
AsterModeEntry
JAS-Provided PreferenceManager
1: editAsterRates():int
1.1: setVisible(boolean):void
1.1.1: getProperty(java.lang.String,java.lang.String):java.lang.String
1.1.2: parseASTERMode():AsterModeEntry
1.1.2.1: <constructor>(List)
2: add():void
2.1: <constructor>(List)
2.2: updateDisplay():void
2.3: writePreferences():void
3: save():void
3.1: writePreferences():void

Figure 6-2: Add ASTER Mode

6.1.3 Edit the ASTER Imaging Modes

Figure 6-3 describes the interactions between classes when editing an ASTER imaging mode.

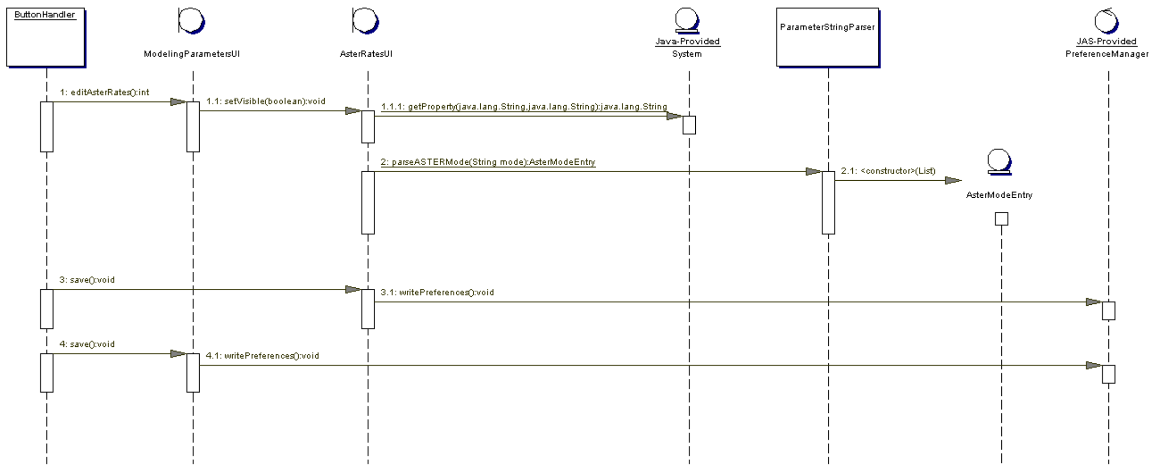


Figure 6-3: Edit an ASTER Mode

6.2 Dump Windows UI

The DumpWindowsUI is a JDialog displayed when the user wants to modify or add dump windows. Through this dialog box the user performs one of the following options: Add a new Dump Window; Edit an existing Dump Window; or delete an existing Dump Window

6.2.1 Dump Windows UI Class Diagram

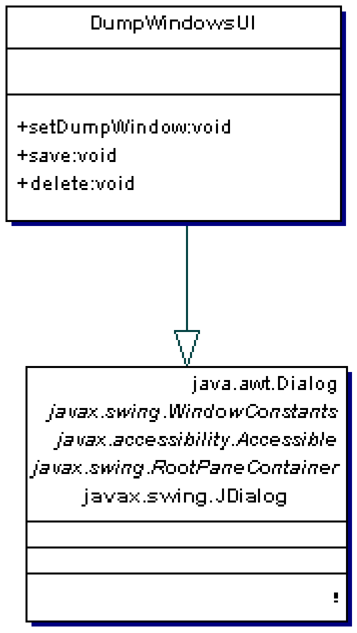


Figure 6-4: DumpWindowsUI class diagram

6.2.1.1 Major Methods

- ***public void setDumpWindow(DumpWindow dump)*** – set the values of the Dump Window to the specified values.
- ***public void save()*** – stores the modified values to the Dump Window.
- ***public void delete()*** – removes the currently displayed Dump Window entry.

6.2.2 Add Dump Window Sequence Diagram

Figure 6-5 describes the interaction between classes when adding a new dump window to a contact.

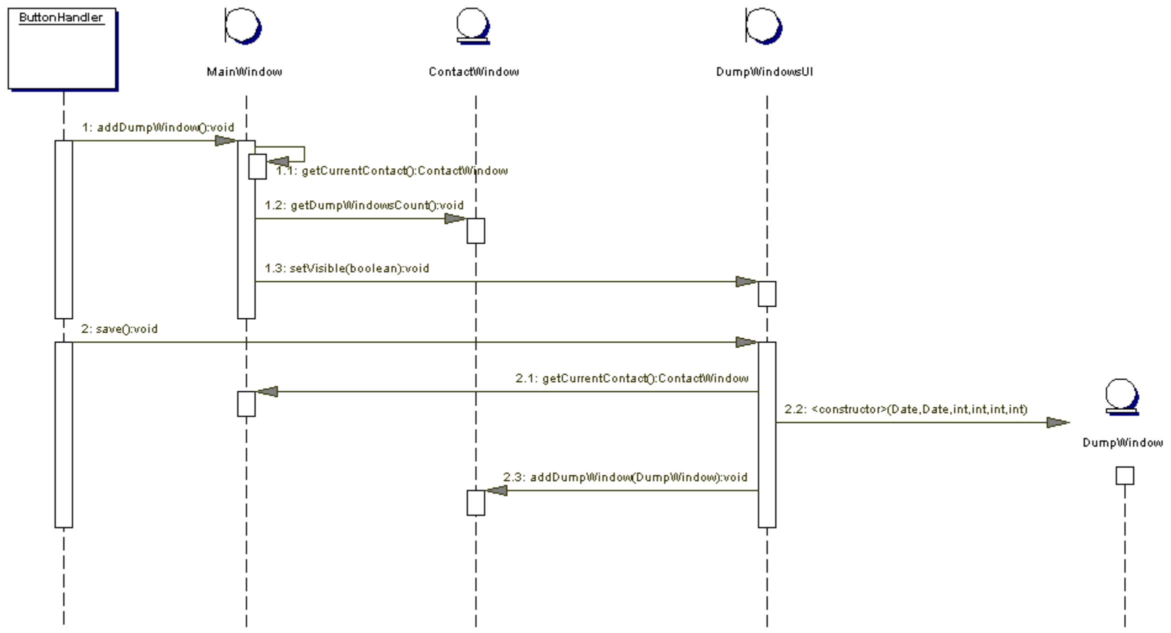

ButtonHandler
MainWindow
ContactWindow
DumpWindowsUI
1: addDumpWindow():void
1.1: getCurrentContact():ContactWindow
1.2: getDumpWindowsCount():void
1.3: setVisible(boolean):void
2: save():void
2.1: getCurrentContact():ContactWindow
2.2: <constructor>(Date,Date,int,int,int,int)
DumpWindow
2.3: addDumpWindow(DumpWindow):void

Figure 6-5: Add a Dump Window

6.2.3 Delete Dump Window Sequence Diagram

Figure 6-6describes the interactions between classes when removing an existing dump window from a contact.

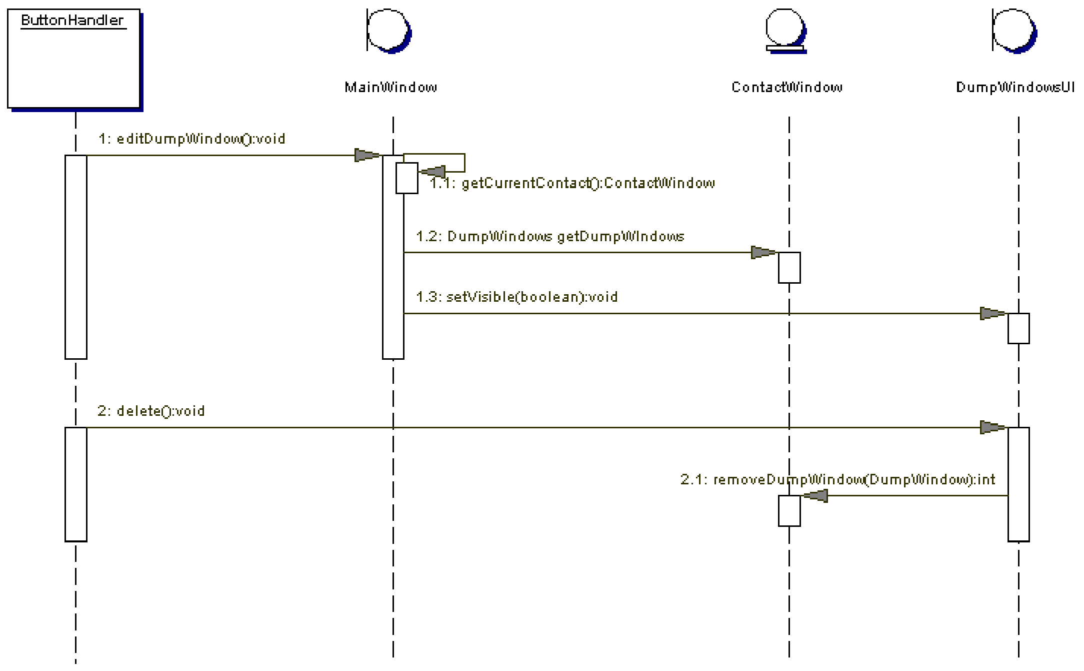

ButtonHandler
MainWindow
ContactWindow
DumpWindowsUI
1: editDumpWindow():void
1.1: getCurrentContact():ContactWindow
1.2: DumpWindows getDumpWIndows
1.3: setVisible(boolean):void
2: delete():void
2.1: removeDumpWindow(DumpWindow):int

Figure 6-6: Delete a Dump Window

6.2.4 Edit Dump Window Sequence Diagram

Figure 6-7 describes the interactions between classes when editing a dump window entry.

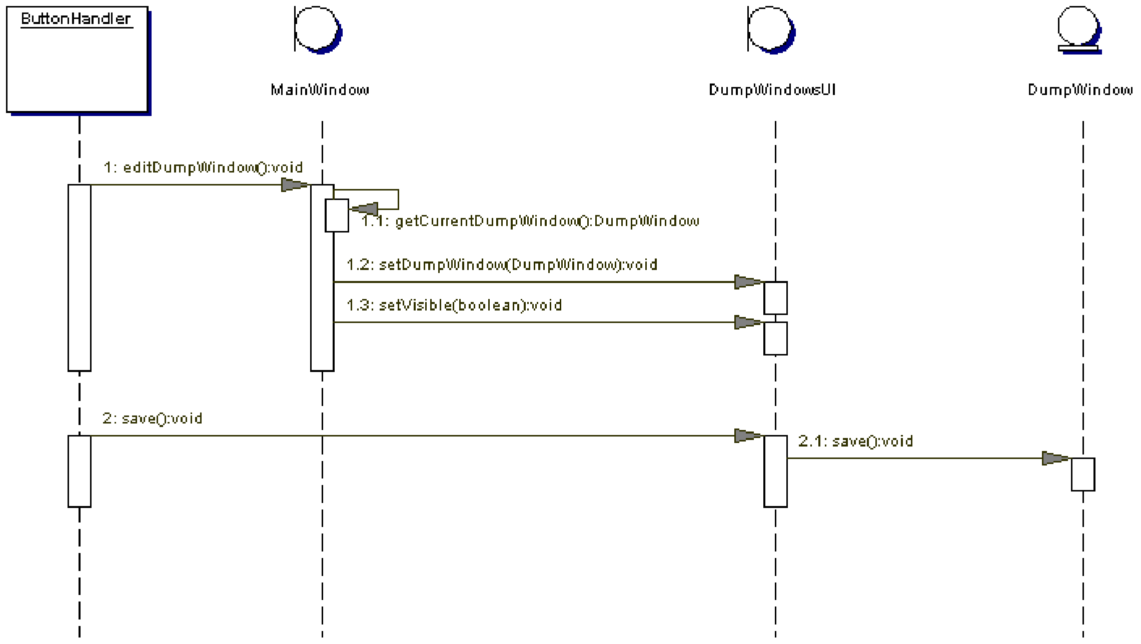

ButtonHandler
MainWindow
DumpWindowsUI
DumpWindow
1: editDumpWindow():void
1.1: getCurrentDumpWindow():DumpWindow
1.2: setDumpWindow(DumpWindow):void
1.3: setVisible(boolean):void
2: save():void
2.1: save():void

Figure 6-7: Edit a Dump Window

6.3 Filter UI

The FilterUI is a configurable dialog that provides a common front end to both print and display filter options. Through this GUI, the user selects which fields and event types to display and/or print depending on which filter type is being edited.

6.3.1 Filter UI Class Diagram

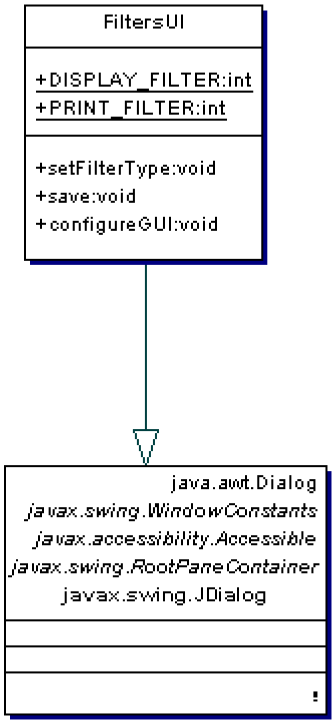


Figure 6-8: FiltersUI class diagram

6.3.1.1 Major Methods

?? ***public void setFilterType(int filterType)*** – sets the type of the specified type (DISPLAY_FILTER or PRINT_FILTER).

?? ***public void save()*** – saves the modified filter settings.

?? ***public void configureGUI()*** – configures the GUI contents based on the type of Filter (print or display).

6.3.2 Edit Display Filters Sequence Diagram

Figure 6-9 describes the interactions between classes when editing the display filters.

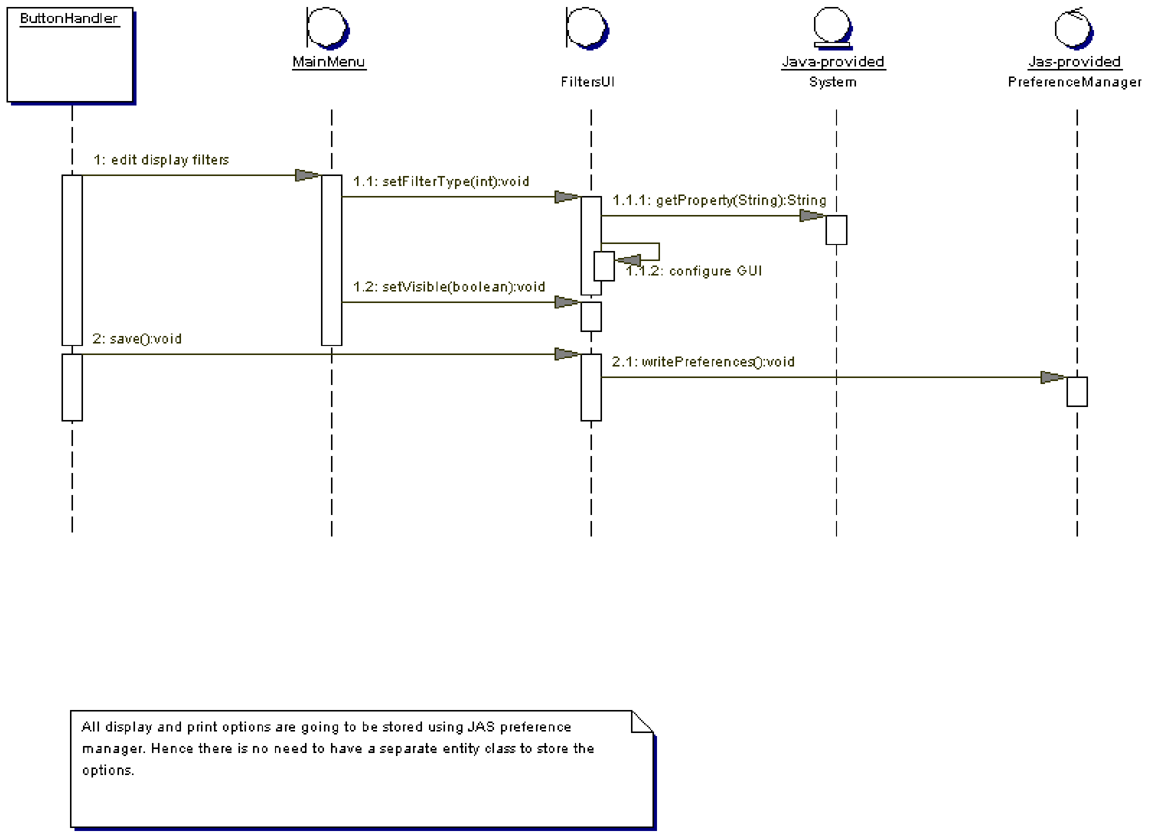
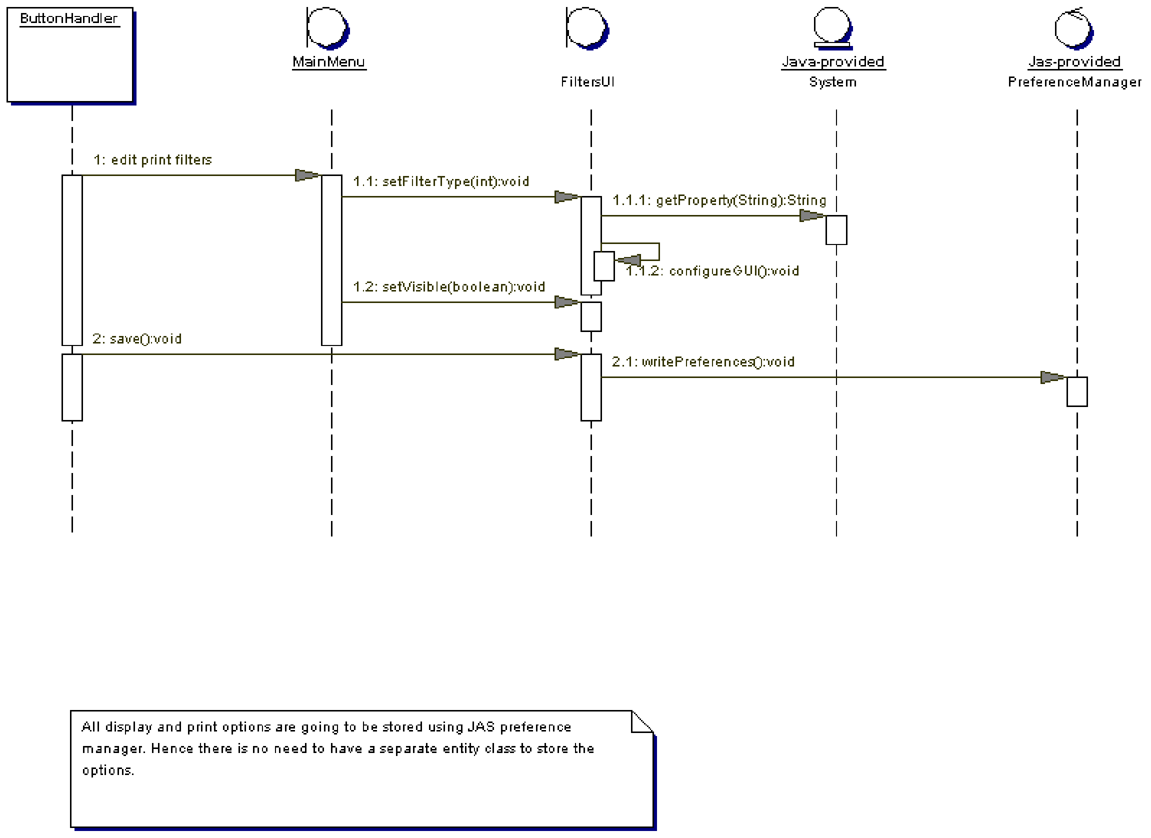

ButtonHandler
MainMenu
FiltersUI
Java-provided
System
Jas-provided
PreferenceManager
1: edit display filters
1.1: setFilterType(int):void
1.1.1: getProperty(String):String
1.1.2: configure GUI
1.2: setVisible(boolean):void
2: save():void
2.1: writePreferences():void
All display and print options are going to be stored using JAS preference manager. Hence there is no need to have a separate entity class to store the options.

Figure 6-9: Edit Display Filters

6.3.3 Edit Print Filters Sequence Diagram

Figure 6-10 describes the interactions among classes when editing the print filters.


ButtonHandler
MainMenu
FiltersUI
Java-provided
System
Jas-provided
PreferenceManager
1: edit print filters
1.1: setFilterType(int):void
1.1.1: getProperty(String):String
1.1.2: configureGUI():void
1.2: setVisible(boolean):void
2: save():void
2.1: writePreferences():void

All display and print options are going to be stored using JAS preference manager. Hence there is no need to have a separate entity class to store the options.

Figure 6-10: Edit Print Filters

6.4 Input Reports UI

The InputReportsUI provides an interface through which the user selects the names and locations of the input reports necessary for schedule generation on the local machine.

6.4.1 Input Reports UI Class Diagram

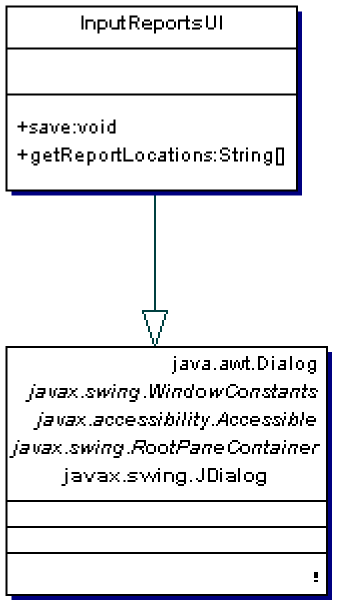

InputReportsUI
+save:void
+getReportLocations:String[]
java.awt.Dialog
javax.swing.WindowConstants
javax.accessibility.Accessible
javax.swing.RootPaneContainer
javax.swing.JDialog

Figure 6-11: InputReportsUI class diagram

6.4.1.1 Major Methods

- ***public String[] getReportLocations()*** – returns an array of strings containing the locations of the input reports.
- ***public void save()*** – saves the user provided/modified report names and locations.

6.4.2 Input Reports UI Sequence Diagram

Figure 6-12 describes the interactions between classes when specifying the names and locations of input reports on the local machine.

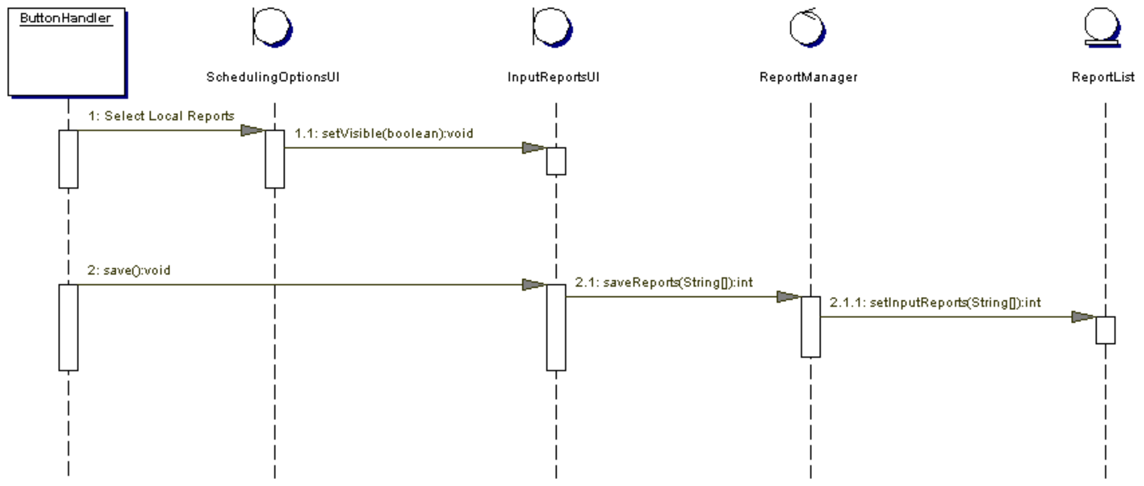


Figure 6-12: Specify Input Reports

6.5 Main Window

The MainWindow is the workhorse class for the GUI thread. Through this class, the user controls schedule generation, enters scheduling parameters, changes modeling parameters, and prints/saves generated playback schedules.

6.5.1 Main Window Class Diagram

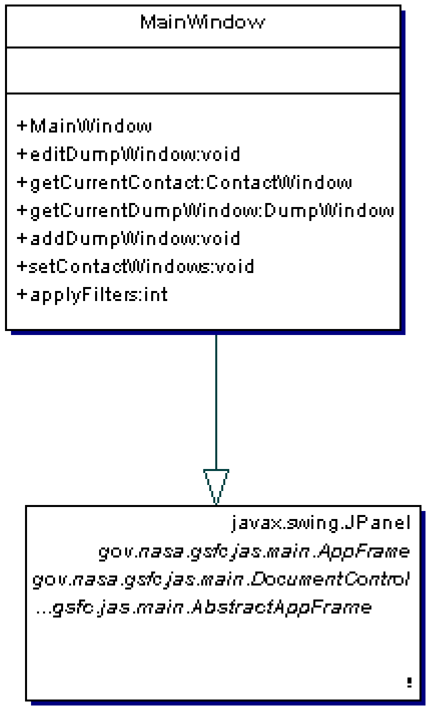

MainWindow
+MainWindow
+editDumpWindow:void
+getCurrentContact:ContactWindow
+getCurrentDumpWindow:DumpWindow
+addDumpWindow:void
+setContactWindows:void
+applyFilters:int
javax.swing.JPanel
gov.nasa.gsfc.jas.main.AppFrame
gov.nasa.gsfc.jas.main.DocumentControl
...gsfc.jas.main.AbstractAppFrame

Figure 6-13: MainWindow class diagram

6.5.1.1 Major Methods

?? ***public void editDumpWindow()*** – edit the selected dump window contents.

?? ***public ContactWindow getCurrentContact()*** – returns the currently selected contact or the contact associated with the selected Dump Window.

?? ***public DumpWindow getCurrentDumpWindow()*** – returns the currently selected Dump Window entry.

?? ***public void addDumpWIndow()*** – add a dump window to the selected contact.

?? ***public void setContactWindows(List contacts)*** – set the contact windows to those specified in the contacts List.

?? ***public int applyFilters()*** – filter the schedule and contact information based on the display filters.

6.6 Parameter UI

The ParameterUI is a dialog through which the user specifies and/or modifies the modeling parameters settings used during playback schedule generation.

6.6.1 Parameter UI Class Diagram

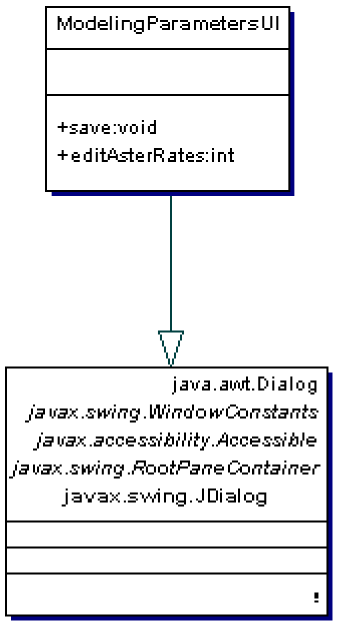

ModelingParametersUI
+save:void
+editAsterRates:int
java.awt.Dialog
javax.swing.WindowConstants
javax.accessibility.Accessible
javax.swing.RootPaneContainer
javax.swing.JDialog

Figure 6-14: ParamaterUI class diagram

6.6.1.1 Major Methods

?? ***public int editAsterRates()*** – modify the ASTER Imaging modes and rates.

?? ***public void save()*** – save the modified modeling parameters.

6.6.2 Parameter UI Sequence Diagram

Figure 6-15 describes the interactions between classes when editing the modeling parameters.

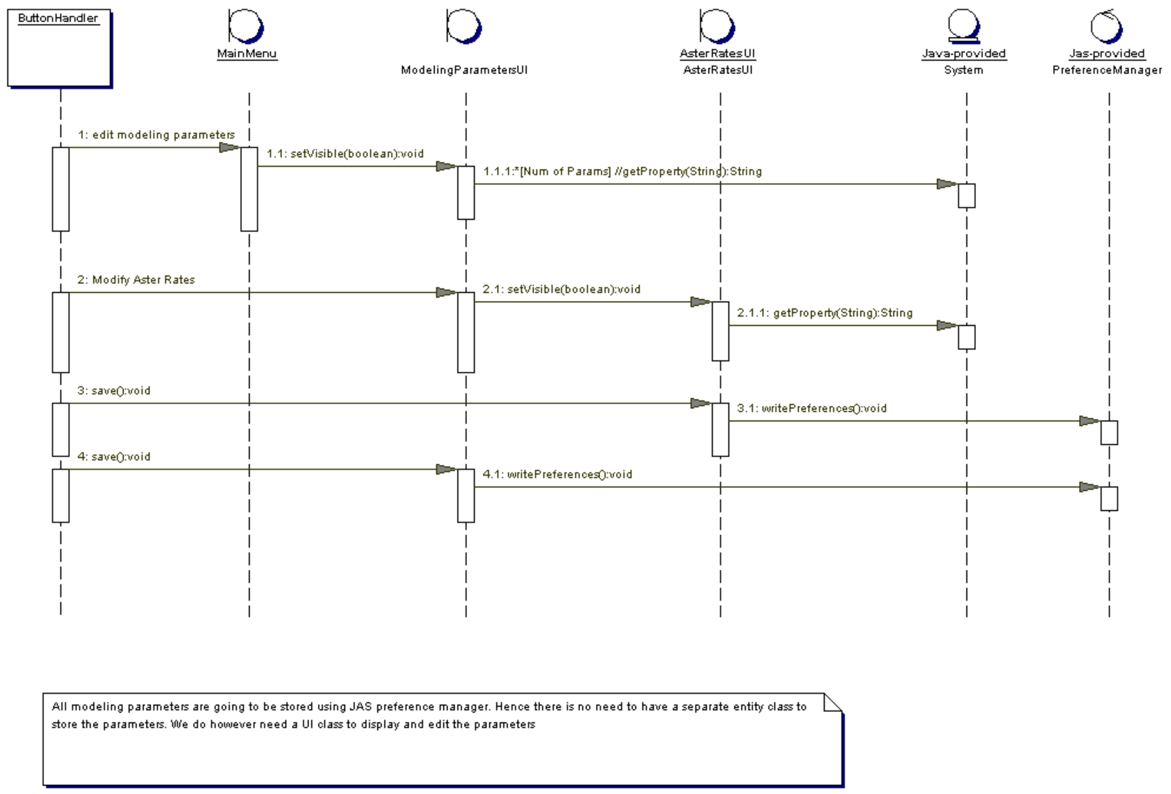

ButtonHandler
MainMenu
ModelingParametersUI
AsterRatesUI
AsterRatesUI
Java-provided
System
Jas-provided
PreferenceManager
1: edit modeling parameters
1.1: setVisible(boolean):void
1.1.1:*[Num of Params] //getProperty(String):String
2: Modify Aster Rates
2.1: setVisible(boolean):void
2.1.1: getProperty(String):String
3: save():void
3.1: writePreferences():void
4: save():void
4.1: writePreferences():void
All modeling parameters are going to be stored using JAS preference manager. Hence there is no need to have a separate entity class to store the parameters. We do however need a UI class to display and edit the parameters

Figure 6-15: Edit Modeling Parameters

6.7 Print Schedule UI

The PrintScheduleUI is a user interface through which the user specifies the printer or file to which the currently displayed schedule will be printed or piped.

6.7.1 Print Schedule UI Class Diagram

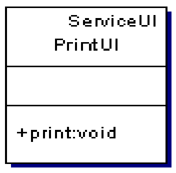


Figure 6-16: PrintScheduleUI class diagram

6.7.1.1 Major Methods

?? ***public void print()*** – print a hardcopy of the current playback schedule.

6.7.2 Print Schedule UI Sequence Diagram

Figure 6-17 describes the interactions between classes when printing a playback schedule.

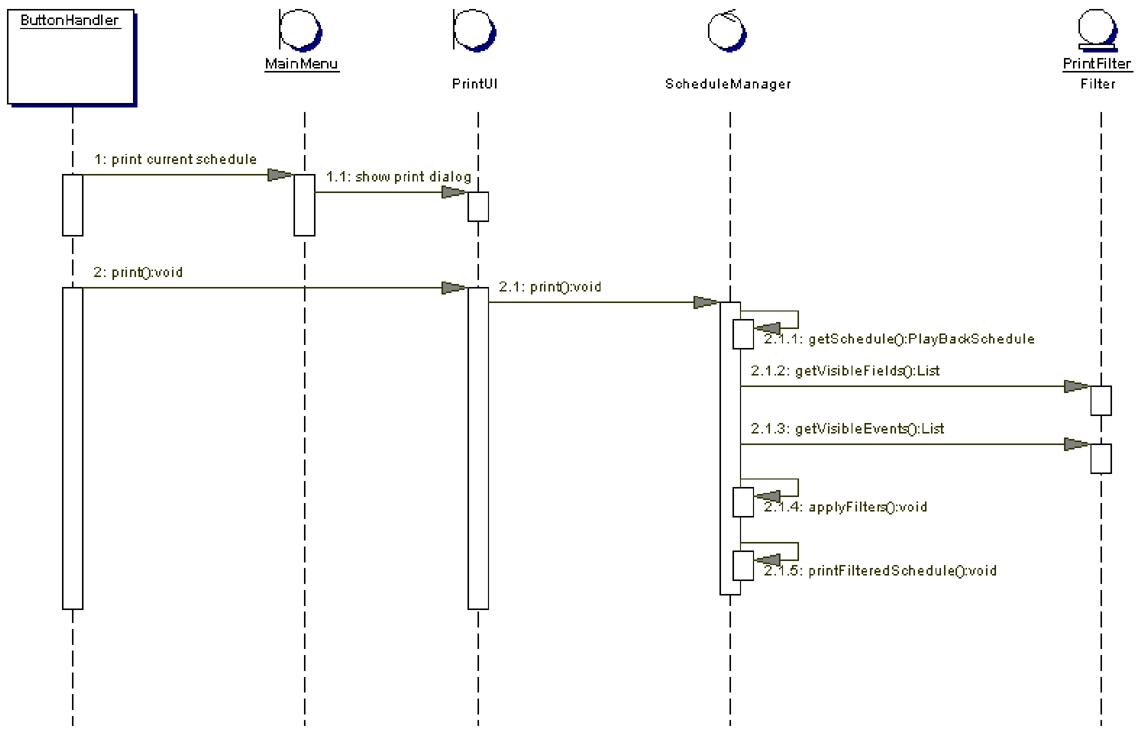

ButtonHandler
MainMenu
PrintUI
ScheduleManager
PrintFilter
Filter
1: print current schedule
1.1: show print dialog
2: print():void
2.1: print():void
2.1.1: getSchedule():PlayBackSchedule
2.1.2: getVisibleFields():List
2.1.3: getVisibleEvents():List
2.1.4: applyFilters():void
2.1.5: printFilteredSchedule():void

Figure 6-17: Print the playback schedule

6.8 Save Schedule UI

The SaveScheduleUI is implemented as a standard File Chooser Dialog (JFileChooser). It is displayed when the operator selects the save schedule option from the Main Menu. Through this UI the user selects the name and location for the file.

6.8.1 Save Schedule UI Sequence Diagram

Figure 6-18 describes the interactions between classes when saving a playback schedule to a file.

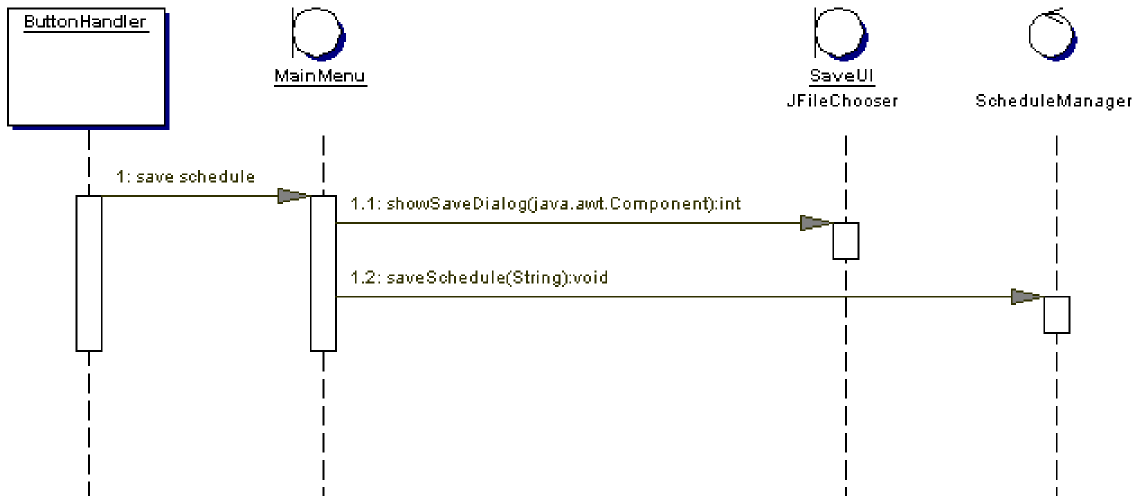

ButtonHandler
MainMenu
SaveUI
JFileChooser
ScheduleManager
1: save schedule
1.1: showSaveDialog(java.awt.Component):int
1.2: saveSchedule(String):void

Figure 6-18: Save a playback schedule

6.9 Scheduling Options UI

The SchedulingOptionsUI is implemented as a JDialog. Through this UI, the user enters and/or modifies the scheduling options. They can set the scheduling window, choose the location and name of the input reports, select the ASTER modeling mode and select whether or not ground contacts are used for playbacks.

6.9.1 Scheduling Options Class Diagram

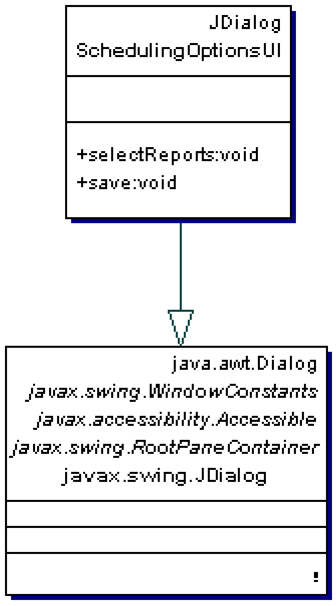


Figure 6-19: SchedulingOptions class diagram

6.9.1.1 Major Methods

?? ***public void selectReports()*** – select the location and name of the input reports on the local machine.

?? ***public void save()*** – save the user modified scheduling options.

6.9.2 Scheduling Options Sequence Diagram

Figure 6-20 describes the interactions between classes when editing the scheduling option values.

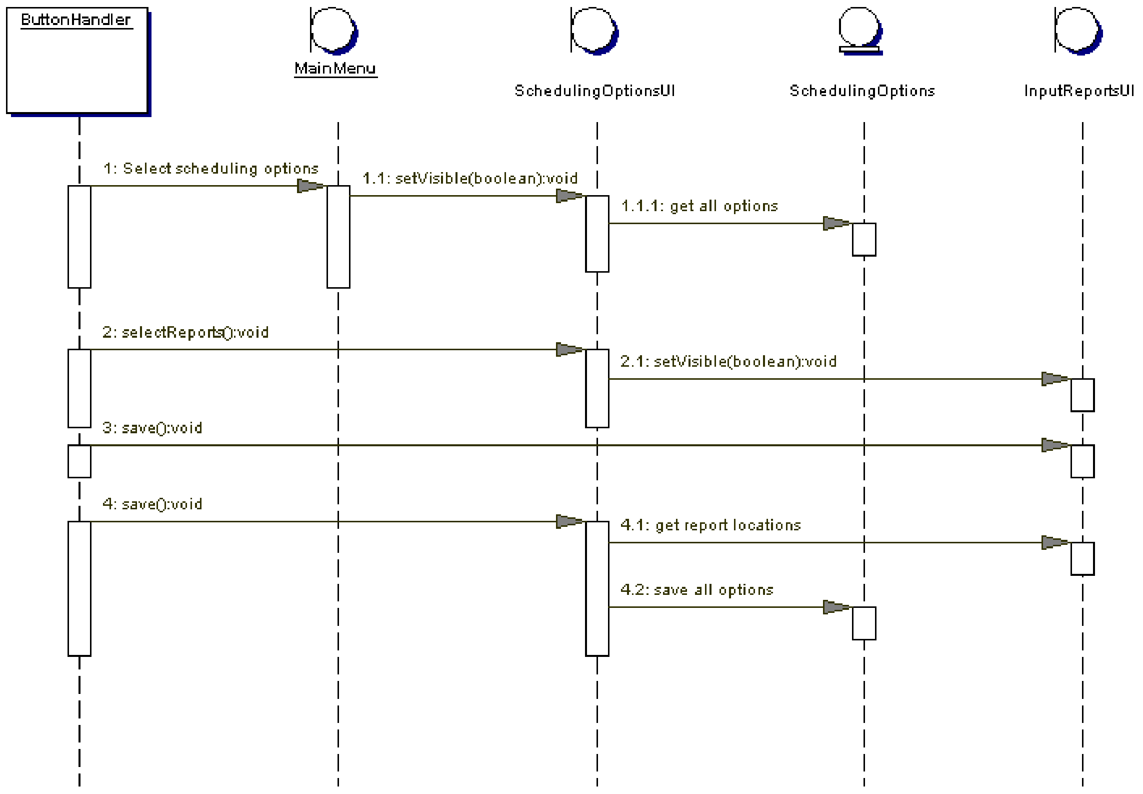


Figure 6-20: Edit Scheduling Options

6.10 Sync Point Parameters UI

The SyncPointParametersUI provides a dialog through which the operator can select a synchronization point. All sync point candidate entries are displayed in this dialog. The current sync point will have the radio button next to the entry selected. The user can change the sync point by clicking on the radio button next to the desired entry.

6.10.1 Sync Point Parameters UI Class Diagram

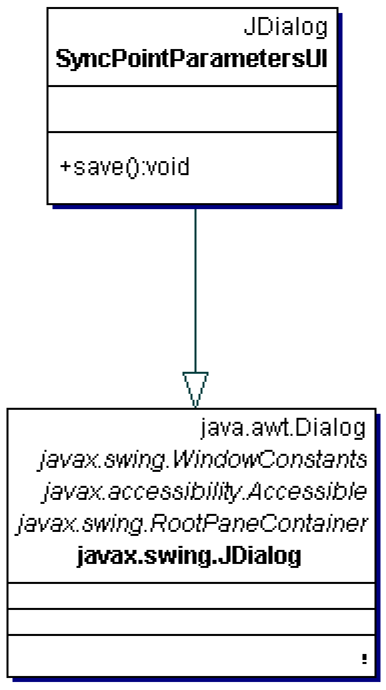

JDialog
SyncPointParametersUI
+save():void
java.awt.Dialog
javax.swing.WindowConstants
javax.accessibility.Accessible
javax.swing.RootPaneContainer
javax.swing.JDialog

Figure 6-21: SyncPointParamatersUI class diagram

6.10.1.1 Major Methods

?? ***public void save()*** – saves the modified sync point parameter values.

6.10.2 Sync Point Parameters UI Sequence Diagram

Figure 6-22 describes the interactions between classes when editing the synchronization point parameters.

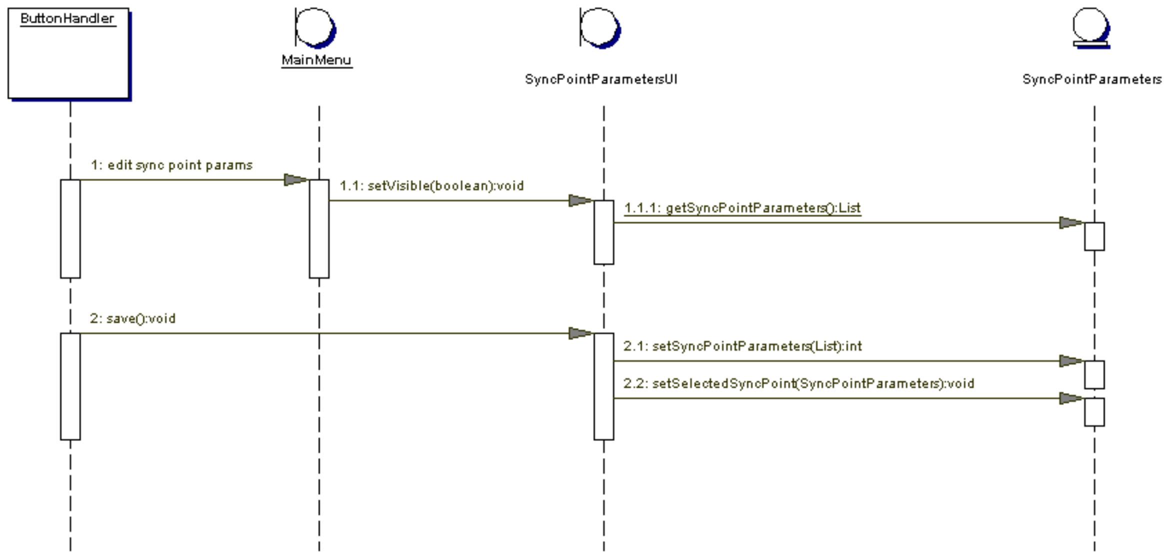
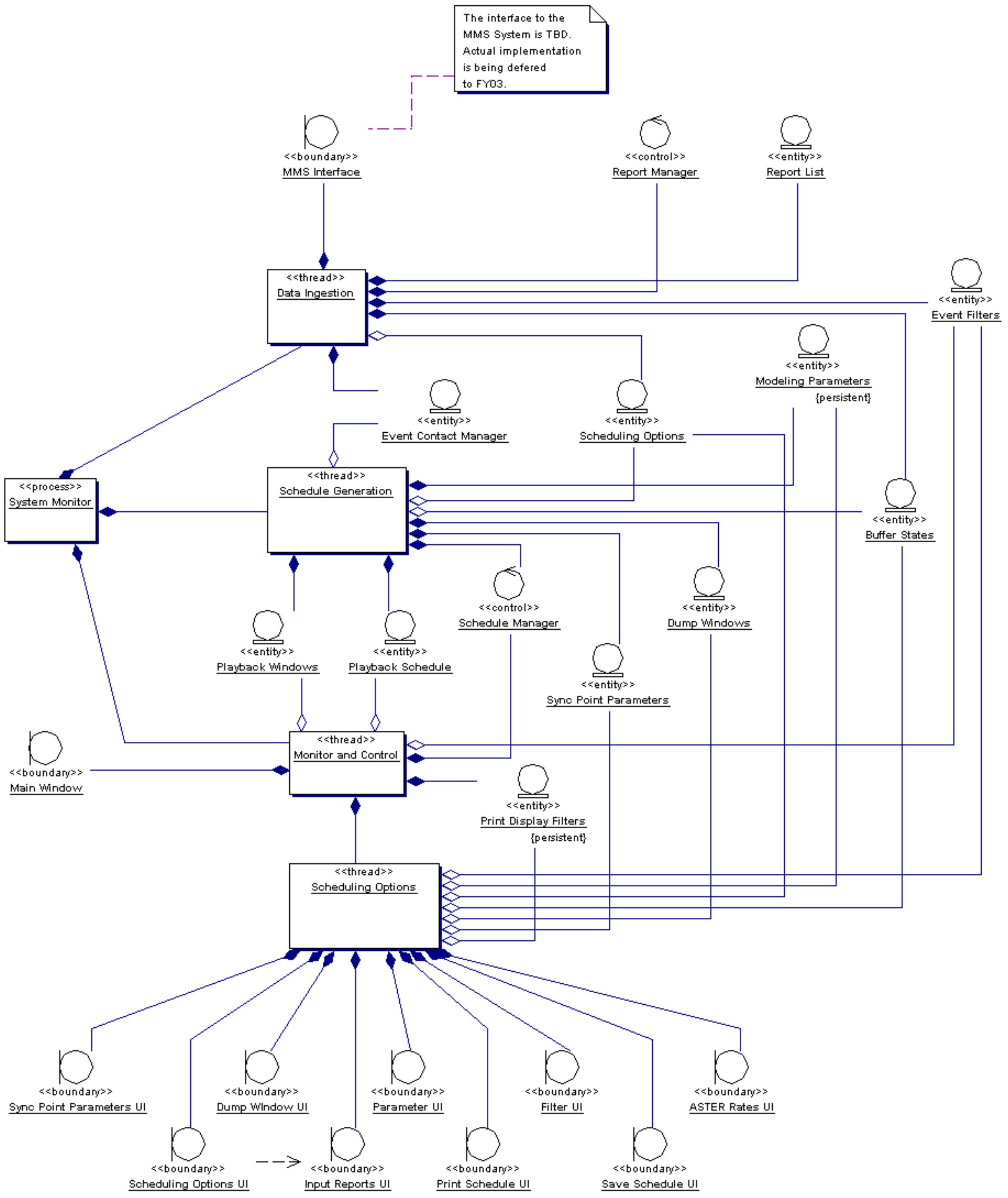

ButtonHandler
MainMenu
SyncPointParametersUI
SyncPointParameters
1: edit sync point params
1.1: setVisible(boolean):void
1.1.1: getSyncPointParameters():List
2: save():void
2.1: setSyncPointParameters(List):int
2.2: setSelectedSyncPoint(SyncPointParameters):void

Figure 6-22: Edit Sync Point parameters

7 Process View

This section presents an architectural view that describes the concurrent aspect of the SPLAT system: tasks (processes and threads), persistent objects and their interactions.

7.1 Processes and Threads


The interface to the MMS System is TBD. Actual implementation is being defered to FY03.
<<boundary>> MMS Interface
<<control>> Report Manager
<<entity>> Report List
<<entity>> Event Filters
<<thread>> Data Ingestion
<<entity>> Modeling Parameters {persistent}
<<entity>> Event Contact Manager
<<entity>> Scheduling Options
<<process>> System Monitor
<<thread>> Schedule Generation
<<entity>> Buffer States
<<control>> Schedule Manager
<<entity>> Dump Windows
<<entity>> Playback Windows
<<entity>> Playback Schedule
<<entity>> Sync Point Parameters
<<boundary>> Main Window
<<thread>> Monitor and Control
<<entity>> Print Display Filters {persistent}
<<thread>> Scheduling Options
<<boundary>> Sync Point Parameters UI
<<boundary>> Dump Window UI
<<boundary>> Parameter UI
<<boundary>> Filter UI
<<boundary>> ASTER Rates UI
<<boundary>> Scheduling Options UI
<<boundary>> Input Reports UI
<<boundary>> Print Schedule UI
<<boundary>> Save Schedule UI

I. Figure 7-1: SPLAT Process View Diagram

8 Size and Performance

This section provides a description of the major dimensioning characteristics of the software that affect the architecture, as well as the target performance constraints.

1. **Average schedule generation times** – These requirements were derived from discussions with the Flight Operations Team (FOT) staff charged with generating special event schedules. Scheduling these events manually can take anywhere from 30 minutes to several days depending on the complexity of the event. Since SPLAT will automate the manual task of generating these schedules, a significant time savings is expected and schedule generation times would be in-line with those mentioned in the requirements specification [1].
2. **Average report extraction times** – These requirements were derived from discussions with the FOT staff. Since the SPLAT feature supporting automated extraction of reports adds only a minimal overhead over the manual procedure, the report extraction time limits are reasonable [1].

9 Quality

The software architecture described will support the reliability and supportability requirements for the system by virtue of the loose coupling between the MMS system (TBD) and the SPLAT tool and the modular design of the system. Usability requirements are addressed by using simple, yet functional, graphical user interfaces for operator interaction with the system.

10 Modeling Limitations/Constraints

The following constraints/limitations are placed on the model of the Terra SSR Buffers in the SPLAT System.

10.1 Events

The following events are parsed from the input reports and used to determine playback schedules:

1. Nadir Term Crossing to Day Events
2. Nadir Term Crossing to Night Events
3. K-Band AOS Events
4. K-Band LOS Events
5. S-Band AOS Events
6. S-Band LOS Events
7. X-Band AOS Events
8. X-Band LOS Events
9. ASTER RTCS Events
10. SSR Buffer States

10.2 Constraints

The following constraints are used during schedule generation:

1. All specified rates are specified in bits/second (bps).
2. The ASTER IMAGING rate will be one of the following:
    a. User selectable percentage – default of zero
    b. Automated – parse ASTER RTCS events from ATC Load Report.
3. The following ASTER RTCS_IDs will be used for scheduling in automated ASTER mode:
    a. RTCS_ID = 5
        1. SWIR turned on 10 seconds after event.
        2. TIR turned on 9 seconds after event.
        3. VNIR1 turned on 11 seconds after event.
        4. VNIR2 turned on 12 seconds after event.
    b. RTCS_ID = 7
        1. TIR is turned off 4 seconds after event.
        2. SWIR is turned off 3 seconds after event.
        3. VNIR1 is turned off 2 seconds after event.
    c. RTCS_ID = 8
        1. TIR is turned on 7 seconds after event.
        2. SWIR is turned on 8 seconds after event.

   d. RTCS_ID = 9
      1. TIR is turned off 3 seconds after event.
      2. SWIR is turned off 2 seconds after event.
   e. RTCS_ID = 13
      1. VNIR 3 is turned off 13 seconds after event.
   f. RTCS_ID = 14
      1. TIR is turned on 2 seconds after event and turned off 32 seconds after event.

4. The imaging rates for the ASTER Instruments are as follows:
   a. VNIR1 – 31,019,000 bps
   b. VNIR2 – 31,019,000 bps
   c. SWIR – 23,053,00 bps.
   d. TIR – 4,109,000 bps
5. The MODIS Day Imaging rate is 10,686,117.6470588 bps.
6. The MODIS Night Imaging rate is 3,192,760.18099548 bps.
7. The MISR Day Imaging rate is 6,486,877.8280543 bps.
8. The MISR Night Imaging rate is 1,013.57466063348 bps.
9. MODIS transition to day mode is keyed to "Nadir Term Crossing To Day" events extracted from the Orbital Events report.
10. MODIS transition to night mode is keyed to "Nadir Term Crossing to Night" event extracted from the Orbital Events report.
11. MISR Transitions to Day 3.5 minutes before MODIS.
12. MISR Transitions to Night 3.5 minutes after MODIS.
13. The K band contact playback rate is 150,000,000 bps.
14. The X-band contact playback rate is 150,000,000 bps.
15. The ASTER Buffer capacity is 84,813,557,760 bits or 58 Supersets.
16. The MISR Buffer capacity is 29,246,054,400 bits or 20 Supersets.
17. The MODIS Buffer capacity is 52,642,897,920 bits or 36 Supersets.
18. The conversion rate for bits to supersets is 1,462,302,720 bits/Superset.
19. The offset between the start of a Dump Window and the start of a playback window will be use configurable with a default value of 0.
20. If an S-band contact starts more than 60 seconds earlier than it's corresponding K-band contact then the Dump window start time is offset 30 seconds after the start of the k-band contact, and the dump window stop time is offset 180 seconds before the end of the K-band contact.
21. If the start time of a S-band contact doesn't precede it's corresponding X-band contact by more than 60 seconds, the dump window start time is offset from the start of the S-band contact by 90 seconds and the dump window stop time is offset by 180 seconds before the end of the k-band contact.

11 Glossary

Table 11-1 contains a list of the acronyms and abbreviations used in this document along with a brief description of each acronym.

Acronym/Abbreviation	**Term**	**Definition**
AOS	Acquisition of Signal	A term describing the acquisition of signal for a TDRS satellite or ground station. Used in determining dump times for Terra.
ASTER	Advanced Spaceborne Thermal Emission and Reflection	Instrument on-board TERRA owned and operated by the Japanese space agency.
ATC	Absolute Time Command	
FOT	Flight Operations Team	The group of engineers charged with monitoring and maintaining a spacecraft on orbit.
GN	Ground Network	Refers to the series of ground stations (Alaska and Norway) to which SSR buffer data may be downlinked.
GOC	Goal Oriented Commanding	The predecessor to SPLAT. GOC was to provide a system that allowed an operator to command a satellite or constellation of satellites using natural language commands and goals.
GSFC	Goddard Space Flight Center	
GUI	Graphical User Interface	A graphical interface (dialogs, etc.) through which a user interfaces (communicates) to a computer system or program.
JAS	Java Application Shell	A framework for building Java applications.
LAN	Local Area Network	
LOS	Loss of Signal	A term describing the loss of signal for a TDRS satellite or ground station. Used in determining dump times for Terra.
MISR	Multi-angle Imaging Spectro-	An instrument on the Terra

	Radiometer	spacecraft
MODIS	Moderate Resolution Imaging Spectrometer	An instrument on the Terra spacecraft.
MMS	Mission Management Software	Unique to EOS, this system is the primary mission planning system for Terra. Among other products, it creates the TDRS Contact Report, and includes basic models for generating command loads.
NASA	National Aeronautics and Space Administration	
RTCS	Real Time Command Sequence	
SPLAT	SSR Playback Automation Tool	The tool being developed to assist with SSR buffer playback scheduling for special events.
SSR	Solid State Recorder	This is Terra's on-board storage device. It operates using buffers wherein data from each instrument (4 buffers total) and housekeeping data are stored for later downlink to a ground station.
SWIR	Short Wave Infrared	An ASTER instrument.
TDRS	Tracking and Data Relay Satellite	A satellite to which the SSR buffer information may be relayed.
TIR	Thermal Infrared	An ASTER instrument
UI	User Interface	Synonymous with GUI.
UML	Unified Modeling Language	
VNIR	Visible and Near Infrared	An ASTER instrument